参赛密码 _____

（由组委会填写）

“华为杯”第十四届中国研究生数学建模竞赛

学　　校		河海大学
参赛队号		10294003
队员姓名	1.	王　旭
	2.	尹紫依
	3.	丁毓蓝

参赛密码 _____

（由组委会填写）

“华为杯”第十四届中国研究生

数学建模竞赛

题 目 构建地下物流系统网络

摘 要：

随着城市货运物流需求的快速增长，构建地下物流系统对于缓解城市交通拥堵、减轻城市污染具有重要的意义。本文针对南京市仙林地区的地下物流运输问题，通过运用圆覆盖算法、基于最近邻的改进算法寻找二级节点，通过 K-means 算法确定一级节点，通过最优化模型构建出地下物流系统网络模型。该模型以缓解交通拥堵、降低物流成本为目标，较为科学合理地设计出网络构成，并综合考虑南京市未来的交通需求增长与系统抗风险能力，设计出地下物流系统的建设时序及动态演进过程。

问题一：根据题意需要将区域交通情况改善为至少基本畅通，我们认为拥堵指数降为 4 即满足条件。根据交通拥堵指数与区域货运总量成正比这一假设，计算出各区域需要转入地下物流系统运输的货运量。根据区域中心点的离散程度及区域整体形状，以 890 区域的左边线为界，将整个区域划分为右侧区域与左侧区域：在右侧区域运用圆覆盖模型，在左侧区域运用基于最近邻的改进算法，最终找出所有节点共 29 个，分别给出位置坐标；接着运用 K-means 聚类算法确定其中一级节点共有 4 个，剩余 25 个为二级节点；最终确定各级节点的服务区域及实际货运量，以及各一级节点的转运率：最小为 58.96%，最大为 68.12%。

问题二：在已确定地下物流节点的基础上，选择合适的地下路线以建立该区域的网络拓扑结构。我们先对各区域全天 OD 流量矩阵进行预处理，得到每天需

要通过地下运输的 OD 流量矩阵。以总成本为目标函数，在满足运输要求的约束下最小化目标函数，通过求解最优解，可以找到最合适的边，进而确定网络的拓扑结构。根据网络拓扑结构可以进一步确定各节点的实际货运量和网络中管道的实际流量，计算出该地下物流网络的建设成本为 597.04 亿元，每天的运输成本为 112.49 万元，折旧费用为 165.84 万元，最后得到每天的总成本为 278.33 万元。

问题三：在问题一和问题二已经确定整个网络结构的情况下，仿真运输过程，定义管道中实际流量与管道最大负载流量的比值为管道利用率；定义各节点实际货物流量与所有同级节点货物流量均值的比值与 1 的差为节点的负荷度。通过分析数据，发现在现有网络下主网（园区-一级节点共同构成的网络）中管道$e_{z_3a_3}$利用率偏高，管道$e_{a_3a_4}$利用率偏低；子网（一级节点与其所包含二级节点构成的网络）中管道$e_{a_{25}a_{26}}$利用率偏高，管道$e_{a_{34}a_{35}}$利用率偏低。节点方面a_{24}, a_{26}负荷度偏高，a_{43}偏低。针对利用率高管道和负荷度高的节点，在其周围适当改设服务节点级别，对利用率较低的管道和负荷度较低的节点不作改变，用作抗风险的备用。经过调整以后网络的运输成本有所下降，并且运输速度显著提高。

问题四：对 ULS 做好顶层设计，使其能够满足该区域未来 30 年交通需求的目标。交通需求每年呈 5%增长，我们求得 30 年后的货运需求为目前货运量的 4.32 倍，为满足需求量，我们提出三项扩容方案，包括将双向双轨通道改建为双向四轨，增设一级节点以及延长班车运营时间。由于每年可建设道路长度大致相等，根据园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道长度之间的比例确定权重，按照权重分配建设年限，计算出三类通道需要的建设年限分别为 2 年，3 年，3 年。

关键词：物流网络 圆覆盖 最近邻改进算法 最优化 负荷度 扩容

目　录

一、问题重述

1.1 研究背景

交通拥堵是世界大城市都遇到的“困局”之一。 在 2015 年荷兰导航经营商 TomTom 发布的全球最拥堵城市排名中，中国大陆有十个城市位列前三十名。据中国交通部 2014 年发布的数据，我国交通拥堵带来的经济损失占城市人口可支配收入的 20%，相当于每年国内生产总值(GDP)损失 5-8%。15 座大城市的居民每天上班比欧洲发达国家多消耗 28.8 亿分钟。大量研究表明:“时走时停”的交通导致原油消耗占世界总消耗量的 20%。高峰期，北京市主干线上 300 万辆机动车拥堵 1 小时所需燃油为 240 万-330 万升。2015 年城市交通规划年会发布数据显示：在石油消费方面，我国交通石油消费比重占到了消费总量的 54%，交通能耗已占全社会总能耗 10%以上，并逐年上升。高能耗也意味着高污染和高排放。

导致城市交通拥堵的主要原因是交通需求激增所带来的地面道路上车辆、车次数量巨增，其中部分是货物物流的需求增长。尽管货车占城市机动车总量的比例不大，但由于货运车辆一般体积较大、载重时行驶较慢，车流中如果混入重型车，会明显降低道路的通行能力，因此，其占用城市道路资源的比例较大。如北京，按常规的车辆换算系数（不同车辆在行驶时占用道路净空间的程度），货运车辆所占用的道路资源达 40%。因此，世界各国都在为解决城市交通和环境问题进行积极探索，而处理好货运交通已成为共识。大量实践证明，仅通过增加地面交通设施来满足不断增长的交通需求，既不科学也不现实，地面道路不可能无限制地增加。因此“统筹规划地上地下空间开发”势在必行，“地下物流系统”正受到越来越多发达国家的重视。

我国人口众多、大城市密集、交通状况不佳已经到了迫切需要改善的程度，并且我国城市地铁网络的建设、高铁公路隧道的建造正大规模地进行，地下空间开发利用的规模和速度已居世界前列，地下工程的技术水平也已基本满足需要，构建地下物流系统网络是建设“地下物流系统”必不可少的关键步骤。

1.2 研究问题

已知南京市仙林地区的交通货运区域划分图和相应的货运 OD（Origin Destination）流量矩阵、各区域中心点及区域面积、各区域交通拥堵系数。发展地下物流的目标是在能够缓解交通拥堵直至交通基本畅通的基础上，降低物流成本。

问题一：地下物流节点选择。地下物流网络由一级、二级节点和节点间地下通道构成。各级节点均与地面衔接并实现多式联运。一级节点与物流园区相连且采用 10 吨的大型车辆地下运输，并可跨区域调运货物，从地面收发货物总量上限为 4000 吨，一级节点之间连通。二级节点从地面收发货物总量上限为 3000 吨，且与非本区域一级节点仅通过本区域一级节点连通，所有节点的服务半径在 3 公里范围内。且近似认为区域交通拥堵指数与区域总货运量（进、出之和）成正比；在满足运输要求前提下，转运率低可减少工作量。根据该区域的实际情况建立该区域节点选择模型，确定该区域地下物流网络节点群。计算结果需要但不限于提交：一、二级节点数及位置、各节点的服务范围（经该节点出、入地面货物的起或讫点形成区域）、各节点实际货运量、各一级节点的转运率。

问题二：设计地下通道网络。在地下物流网络节点群的基础上，选择合适的地下路线以建立该区域的“地下物流系统”网络。在转运率变化不大的情况下，可适当调整一、二级节点位置来优化网络。除园区至一级节点的地下通道外其他地下通道均采用 5 吨的地下运输车辆。计算结果需要但不限于提交：网络构成（节点及通道位置）；各节点实际货运量；各级通道的位置和实际流量。

问题三：网络改进。从全局出发，根据对运行情况的仿真，通过增加、减少节点的个数，调整节点的位置或级别，增加、减少、改变路径的方法缩短货物运输总里程，同时节省运输时间，降低运输成本。进一步从增强 ULS 的抗风险能力的角度考虑，例如如某通道中断，某方向货运量激增等情况，对第二问模拟出的 ULS 作出相应的改动。

问题四：建设时序与动态优化。对 ULS 做好顶层设计，使其能够满足该市近 30 年内每年需求量呈 5%增长的交通需求的目标。根据建设进度需要分八年完成“地下物流网络系统”的建设，每年可建设道路长度大致相等，对该市“地下物流系统”网络各线路的建设时序及演进过程进行规划，对比出与第三题中设置的网络的差别，并比较两者的优劣。

二、问题分析

地下物流系统网络的构建近几年来一度成为学术界讨论的热点问题，因其与我们的生活息息相关，所以，合理地规划地下物流系统也具有很强的现实意义。对各物流节点的选址，各节点间的通道路线设计和实际流量以及建造与运输总成本等数据进行分析和建模对于制定合理的地下物流系统具有重要的参考意义。

2.1 问题一的分析

问题一是根据南京市仙林地区的实际情况建立该区域的地下物流节点选择模型，要求确定一、二级节点数量及位置，各节点的服务范围以及实际货运量，同时计算各一级节点的转运率。

首先，根据各区域交通拥堵指数与区域总货运量之间的正比关系，计算得出各区域需要从地下运输的货运量；

其次，将全部区域以 890 区域左侧边线为界划分为两个区域，右侧区域参照圆覆盖模型，左侧区域根据最近邻规则改进算法，建立模型，通过计算确定所有节点的位置及服务范围；

再次，根据 k-means 聚类算法，以降低各一级节点的转运率为目标，建立目标函数划分一级区域，从所有节点中确定一级节点，则其余一级区域内的其他节点则为该区域的二级节点；

最后，计算各节点的地面收发货物总量、实际货运量以及各一级节点转运率。

2.2 问题二的分析

问题二需要在问题一确定的节点基础上构建地下通道网络，使得总成本最小。我们首先对各区域全天 OD 流量矩阵进行预处理，得到每天需要通过地下运输的 OD 流量矩阵。

地下流量矩阵的确定可以进一步计算出各节点间的收发需求量，而在网络节

点已知情况下确定优化的网络拓扑结构就是要寻找出合适的边。同时题目要求总成本要尽可能小，于是我们以总成本为目标函数，在满足运输需求的约束下最小化目标函数。而满足运输需求的约束便是寻找合适边的过程，所以通过求解最优解，可以找到最合适的边进而确定网络的拓扑结构，根据网络拓扑结构可以进一步确定各节点的实际货运量和网络中管道的实际流量，并能给出相关成本。

2.3 问题三的分析

在问题一确定节点，问题二确定网络拓扑的情况下，已经完全确定网络的结构和网络构建时轨道类型的选择。各设计轨道上实际货运流量与轨道最高承载量的比值可以视作该条轨道的使用率。各节点实际货物流量与所有同级节点货物流量均值的比值与 1 的差为节点的负荷度。

对于使用率过高的轨道和负荷度高的节点，我们认为其抗风险能力和发展预留量过少，可以通过增加周围轨道铺设或增设节点来改善状况；对于使用率较低的轨道与负荷度较低的节点，可以考虑作为整个网络的备用空间，不加以改变。

2.4 问题四的分析

问题四是需要对地下物流系统做好顶层设计，考虑该市 30 年内的交通需求变化，给出分八年完成 ULS 的建设时序及演进过程。根据需求量的增长率求得 30 年后的需求量，大概为目前货物需求量的 4 倍，根据货运量的增加考虑对 ULS 进行扩容，可以提出几项扩容方案。由于每年可建设道路长度大致相等，因此需要通过计算各节点间的通道长度确定总工作量，并将所有通道分为园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道三大类，计算出三类通道总长度的比例，确定需要的建设年限权重，根据权重分配建设年限，得到 ULS 线网的大致建设时序。

三、基本假设

1. 假定运行车辆平均速度为供参考的车辆运行速度，不考虑加速度；
2. 设置运输轨道时，节点之间采用直线连接，不考虑转弯部分；
3. 通道与节点每天的折旧率根据会计准则按照每年 360 天进行计算；
4. 货运成本中不考虑车辆消耗电量造成的费用

四、符号说明

参数符号	符号含义
$Z=\{z_1,z_2,z_3,z_4\}$	物流园区点
$A=\{a_1,a_2,\cdots a_m\}$	一级节点
$A'=\{a_{11},a_{12},\cdots a_{1n_1},a_{21},a_{22}\cdots a_{2n_2},\cdots a_{ij}\cdots a_{mn_m}\}$	第i个一级节点下的第j个二级节点
$R=\{e_{ij}\}$	i节点到j节点的轨道
$L=\{l_{ij}\}$	i节点到j节点的轨道里程
$V=\{v_{ij}\}$	i节点到j节点的运货量
TC	网络设计总成本
DC	网络设计折旧费用
CC	网络设计建设成本
GC	网络设计通道成本
TRC	网络设计运输成本
λ	圆覆盖模型中向量的步长值
Θ	圆覆盖模型中向量的偏移角度
τ	节点负荷度
σ	管道利用率

五、模型的建立与求解

5.1 问题一的模型建立与求解

5.1.1 数据的预处理

地下物流网络的第一个直接目标，缓解交通拥堵指数直至交通通畅，至少是基本通畅，我们假定在理想化情况下，不存在交通中断、货运量激增等意外情况发生，即满足交通基本通畅则可以达到目标。此时，参照交通拥堵指数的取值范围及划分，我们确定将各区域的交通拥堵指数降至 4。

我们对附件中给出的全天 OD 流量横纵分别加总，计算出各个区域的进、出货总量，从而分别求得每个区域的总货运量 Q_i，将其与各区域的交通拥堵指数 TSI_i 相对应，得到总货运量与交通拥堵指数之间的比例系数 k_i，具体公式如下：

$$Q_i=TSI_i*k_i$$

因此，我们要将各区域的 TSI_i 降至 4，则可以得到需要转入地下物流系统进行运输的货运量，具体公式如下：

$$\Delta Q_i=Q_i*(TSI_i-4)/TSI_i*k_i$$

由于 895、866、896 三个区域的交通拥堵指数已经小于 4，即已经达到所要求的交通畅通或基本畅通的目标，因此对于这三个区域，我们认为这些区域的货物可以全部通过地上交通运输，不需要将部分货物转入地下运输，我们在后续考虑地下物流节点的选取时不考虑这三个区域。

通过以上公式分别计算得到其他 107 个区域地面运输需要减少的货运量，即需要通过地下运输的货运量ΔQ_i。

由题目要求可知，部分区域货运量与面积之比过小，此类区域当节点覆盖区域中心点即可视为对该区域进行了覆盖。我们将货运量与面积之比简称为量面比，

对所有区域的量面比进行计算，并用图表显示其分布，得到如下图表：

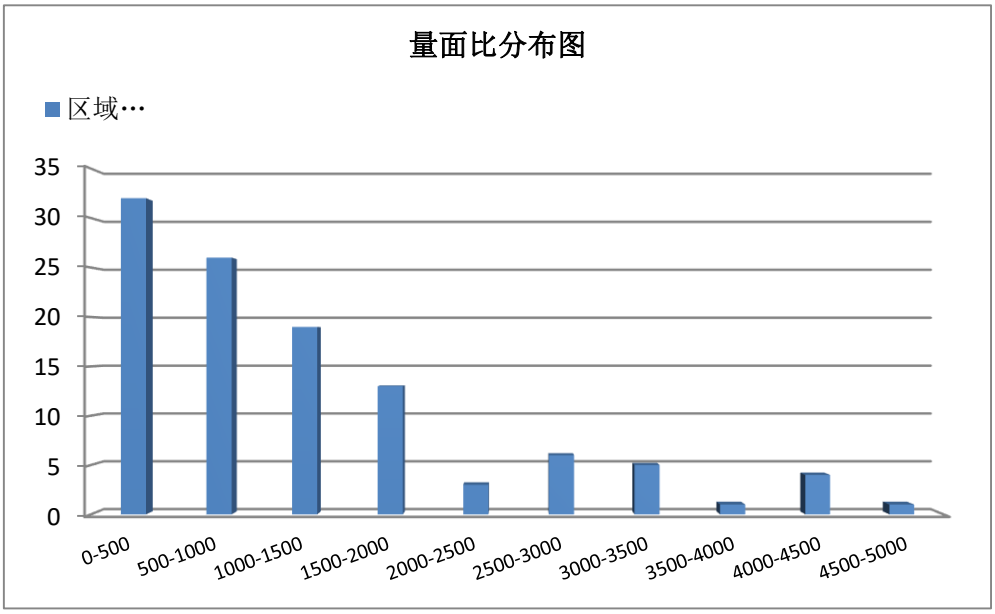


图 5-1 各区域量面比分布图

结合上图，量面比超过 2000（吨/平方千米）的区域数量非常少，根据计算结果，我们求得所有区域量面比的平均值为 1280.004386（吨/平方千米），因此为简化计算，我们假定量面比的正常范围处于 500-2000（吨/平方千米）的范畴内，超过 2000（吨/平方千米）的区域认为其量面比非常高，而在 500（吨/平方千米）以下的区域则认为是量面比过小。

于是，我们将表格中量面比低于 500 的区域与地图中的区域一一对应，用黄色标记出低量面比的区域，如下图所示：

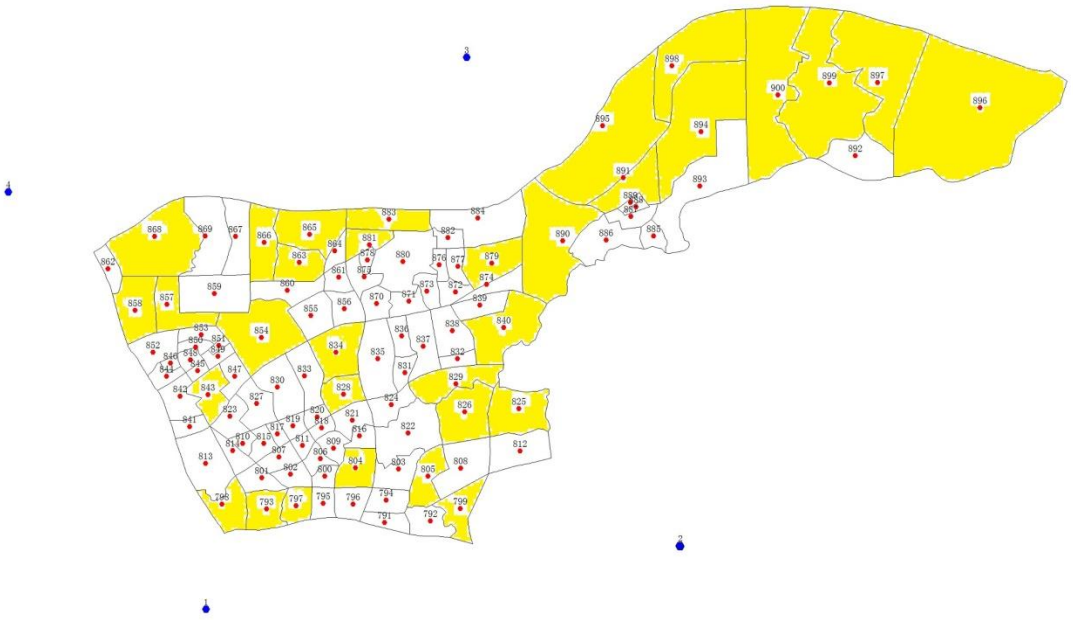


图 5-2 低量面比区域分布图

最后，根据原始地图给出的各区域中心点的坐标，画出的散点分布图在原始坐标系中分布不均。因此，为使得各点均匀分布在坐标系中，同时方便确定节点坐标，我们建立新的坐标轴，分别选择各区域中心点横纵坐标最小的点所在的位置建立坐标轴，即以 1 物流园区点所在水平线作为 x 轴，4 物流园区点所在垂直线作为 y 轴，它们的连线交点为新的坐标原点。根据题目给出的各区域中心点位置坐标确定各中心点在新的坐标轴中的位置，得到中心点分布图如下：

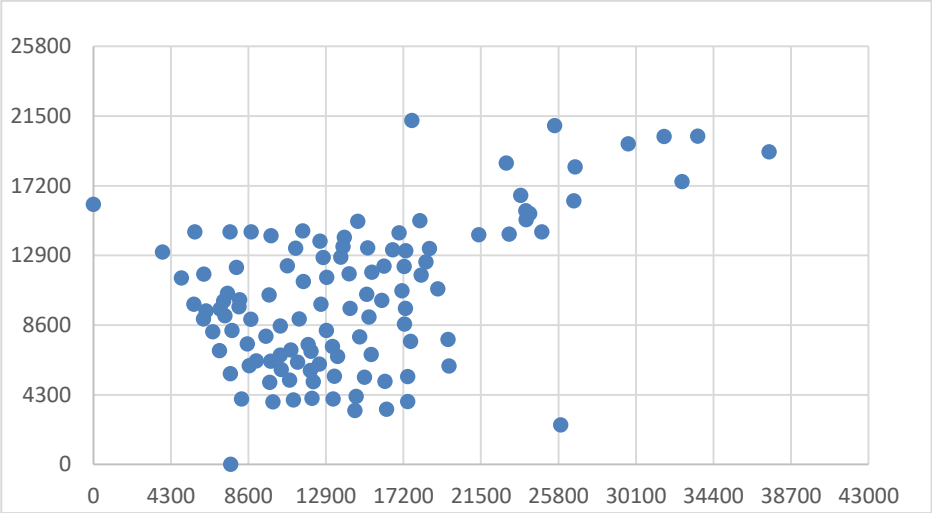


图 5-3 新坐标轴中各区域中心点分布图

5.1.2 圆覆盖算法及最近邻规则改进算法确定节点位置

首先介绍圆覆盖模型算法思想：考虑到节点的服务范围为周围 3000m，即可以认为节点确定了一个以自身坐标为圆心，半径为 3000m 的圆。确定一个圆以后，计算其所覆盖区域中心点的收发货物总量，值小于等于 3000 吨，则把该区域作为这个节点的服务区域；否则，按规则移动圆重复上述步骤，直至所有区域均被纳入服务覆盖。

记待覆盖区域中心点集合为 Rest_set，初始为所有区域中心点 $p_i\ (x_i, y_i)$ 构成；记已覆盖区域中心点集合为 Included_set，初始为空；记圆从本次移动到下次移动这个过程中覆盖的区域中心点集合为 Temp_included_set，初始为空；记算法返回结果集合为 Result_set，集合元素是（圆心，所覆盖的区域编号，覆盖区域货物量）构成的结构体。

算法 5.1 （圆覆盖算法）

输入：初始圆心坐标 $p(x_0, y_0)$，圆的半径 R

输出：Result_set

步骤：

（1） 当 Rest_set 非空时，进入步骤（2），否则，返回 Result_set 并终止。

（2） 遍历 Rest_set 集合中所有元素，计算元素 $p_i\ (x_i, y_i)$ 到圆心 $p(x_0, y_0)$ 的距离D_i

IF $D_i < $ R：

将 p_i 放入 Temp_included_set。

（3） 遍历 Temp_included_set 集合中所有元素，计算其货物量的总和 Total

IF 2500<Total<3000：

对 Temp_included_set 集合所有元素计算一个 Result_set 的结构体

将 Temp_included_set 集合所有元素放入 Included_set

将 Temp_included_set 集合所有元素从 Rest_set 集合删除

置空 Temp_included_set 集合，回到步骤（1）

ELSE：

置空 Temp_included_set 集合，进入步骤（4）

（4） 按规则更新圆心 $p(x_0, y_0)$，进入步骤（1）

下面详细介绍更新圆心的规则。

直观地，为了让圆的移动覆盖更多的点，应该让圆朝着未覆盖点的大体方向上移动，当然同时要让其在这个方向上能有一定的随机扰动范围，已达到更好的覆盖效果。所以，我们计算圆心到 Rest_set 集合中所有元素向量的合成向量为移动方向。由初等数学知识可得合成向量

$$\vec{v} = (x, y) = (\sum_{i=1}^{n}(x_i - nx_0), \sum_{i=1}^{n}(y_i - ny_0))$$

其中，x_i 与 y_i 分别是p_i 的横纵坐标，$p_i \in$ Rest_set，n表示 Rest_set 中元素的个数。

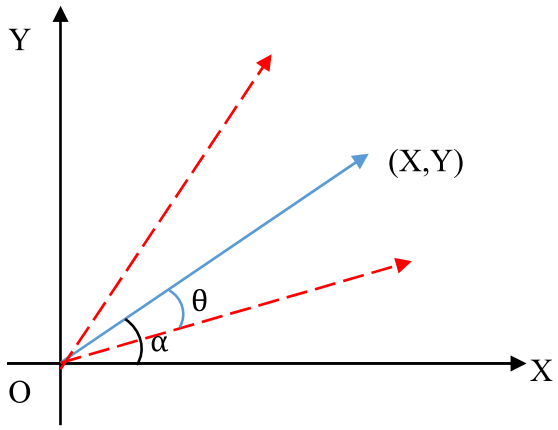
对 $\vec{v}$ 进行单位化，记为(X′, Y′)=

$$\left(\frac{\sum_{i=1}^{n}(x_i - nx_c)}{\sum_{i=1}^{n}(x_i - nx_c)^2 + \sum_{i=1}^{n}(y_i - ny_c)^2}, \frac{\sum_{i=1}^{n}(y_i - ny_c)}{\sum_{i=1}^{n}(x_i - nx_c)^2 + \sum_{i=1}^{n}(y_i - ny_c)^2}\right)$$

如图所示，为使调整方向有一定随机性，考虑加入正负 θ 角的偏移，故最终随机偏移后的向量方向位于图中两个红色向量之间，即新的方向向量与水平方向夹角 $\beta \in [\alpha - \theta,\ \alpha + \theta]$

显然，(X′, Y′)=$(\cos\alpha, \sin\alpha)$，则 $\alpha = \cos^{-1} X'$，故可以令$\beta = \alpha + \mu\theta(-1)^k$，其中 k 是随机生成的奇偶数，以控制偏移方向，μ 是介于[0, 1]的随机数。记超参数 λ 为圆心移动的步长。因此，新的圆心坐标可以表示为：（如下图所示）

$$(x_0 + \lambda\cos\beta, y_0 + \lambda\sin\beta)$$



将该算法应用于图中发现，圆心的移动方向较为固定，无法大量覆盖运输区域。其圆心移动更新如图所示：

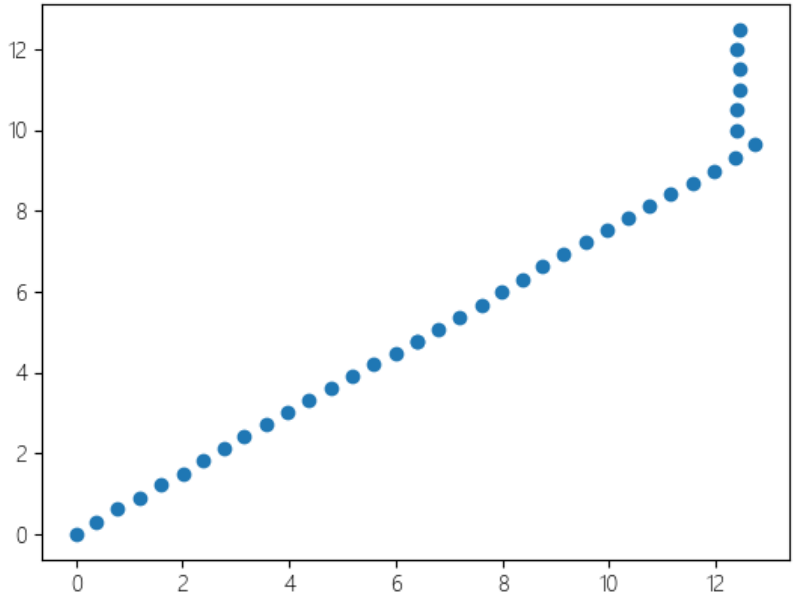


图 5-4 圆覆盖法在左下区域的圆心迭代过程图

理论分析可知，因为左下区域的点非常密集，从初始（0，0）开始，其大体移动方向便是向右上角，因此要想避免这一现象，需要在横纵两个方向上均再加随机干扰，但如此一来圆心的迭代便会是在该区域内无规则的乱跳。再加上，左下区域中心点密集，半径 3000m 的圆可以覆盖较多的点，而这些区域点的货物量累加之和远大于节点的服务能力，因此我们认为圆覆盖模型不适用于左下区域。而右上方整体上呈现狭长形状，且区域中心点的分布相较于左下区域更为稀疏，比较适合该模型。

故而根据区域中心点的离散程度及区域整体形状，以 890 区域的左边线为界，将整个区域划分为左侧区域与右侧区域。在右侧区域运用上述圆覆盖模型，对左侧区域提出其他更适应的模型。

对于右侧区域：

我们以 879 区域中心点为初始圆心，向右搜索覆盖。由于右侧区域呈近似横向长条形状，我们希望圆心的移动是这样的：横向上，始终向右；纵向上，能够上下波动以探索更可能地覆盖区域中心点，而为了使纵向上的上下波动更邮箱，横向上移动应该更加缓慢，因此将横纵坐标的移动步长分开设置，并使得纵向上有向相反方向移动的随机性，我们调整优化圆心的更新规则。

首先，为使得横向移动慢，横坐标的步长设置为$\frac{\lambda}{10}$，纵坐标步长保持为λ。其次，为使得纵向上能够上下改变方向，在纵坐标更新时随机地乘以-1。因此，移动后的新坐标可以表示为：

$$\left(x_0+\frac{\lambda}{10}\cos\beta\,,y_0+\lambda\sin\beta\,(-1)^l\right)$$

其中 l 是随机的奇偶数。

将改进后的模型运用于所选右侧区域，其圆心移动方向情况如图所示：

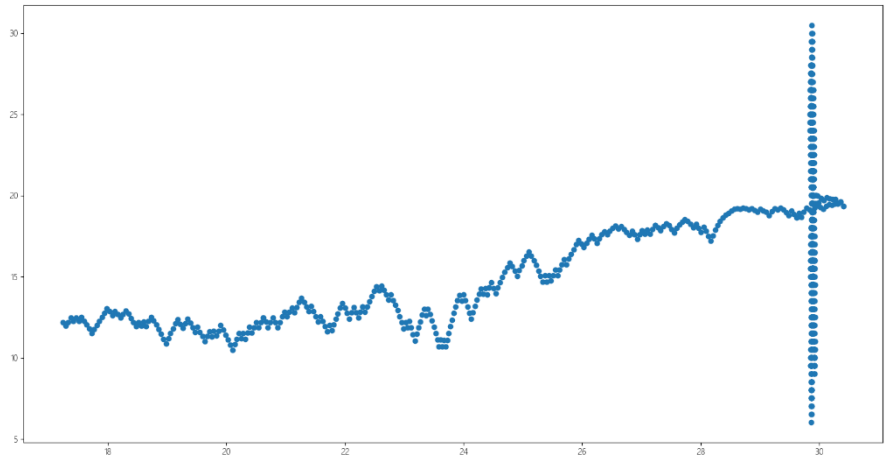


图 5-5 改进的圆覆盖模型在右侧区域圆心移动过程图

模型实际运用后，确定右侧区域共有 6 个节点。

对于左侧区域：

我们基于最近邻算法的原则，提出最近邻改进算法，其思想如下：从某一个区域中心出发，计算所有还未被服务的区域中心到它的距离，将距离小于节点可以服务半径中的最短距离找出，该距离对应的区域中心点纳入临时覆盖集合，更新临时覆盖集合的质心并计算这个集合内区域的货运量，如果新纳入的区域中心对临时覆盖集合的构成的子网是正激励，则留用，如果是负激励则按规则作出相应的处理。

记待覆盖区域中心点集合为 Rest_set，初始为所有区域中心点 $p_i\ (x_i, y_i)$ 构成；记已覆盖区域中心点集合为 Included_set，初始为空；记一次计算过程中覆盖的区域中心点集合为 Temp_included_set，初始为空，开始计算会是首先选定的此次计算出发点先入集合；记一次计算过程未覆盖区域中心点集合为 Temp_rest_set，初始为上一次计算后的 Rest_set；记算法返回结果集合为 Result_set，集合元素是（圆心，所覆盖的区域编号，覆盖区域货物量）构成的结构体。

算法 5.2（基于最近邻的改进算法）

输入：Rest_set

输出：Result_set

步骤：

（1） 当 Rest_set 非空时，进入步骤（2），否则，返回 Result_set，并终止。

（2） 将 Rest_set 的第一个元素弹出，放入 Temp_included_set，并记这个元素的坐标为子网质心坐标$p_0(x_0, y_0)$，计算这时 Temp_included_set 集合元素的总货运量V_{total}

（3） IF $V_{total} <$ 节点吞吐能力：

1) 遍历 Temp_rest_set 集合，找出距离质心$p_0(x_0, y_0)$最近的元素 $p_i(x_i, y_i)$,将其放入 Temp_included_set

2) 临时更新质心为$p_{temp}(x_{temp}, y_{temp})$

3) 遍历 Temp_included_set，计算是否有元素在质心更新后，溢出服务范围，记溢出元素个数为 n

4) $Case\ n=0$：

IF $V_{total}+p_i$的货物量 <= 节点吞吐能力：
从 Temp_rest_set 中移除p_i
V_{total}更新为$V_{total}+p_i$的货物量

更新子网质心$(x_0,y_0)=(x_{temp},y_{temp})$

ELSE：
从 Temp_included_set 中移除p_i
跳出至上层步骤（3）

$Case\ n=1$：

IF 溢出点货物量 > p_i的货物量：
从 Temp_included_set 中移除p_i
ELSE：

更新子网质心$(x_0,y_0)=(x_{temp},y_{temp})$

$Case\ n>1$：

从 Temp_included_set 中移除p_i

（4） 遍历 Temp_included_set 计算一个 Result_set 的结构体元素，计算结果放入 Result_set

（5） 将 Temp_included_set 中的所有元素从 Rest_set 中移除

上述算法步骤中，临时质心的计算方式是：

$$(x_{temp},y_{temp})=\left(\frac{\sum_{i=1}^{n}x_i}{n},\frac{\sum_{i=1}^{n}y_i}{n}\right)$$

其中，x_i与y_i 分别是p_i 的横纵坐标，p_i ∈ Temp_included_set，n表示 Temp_included_set 中元素的个数。

此外，算法中关于更换质心后，对不同个数溢出情况的处理解释如下：如果新加入的元素导致两个及两个以上的原元素溢出，我们简单地认为这个新元素是负激励的，不应该纳入，应重新计算；而如果只导致一个元素溢出，那么我们判定溢出元素和进入元素谁的货运量更大，新元素大就认为它是正激励的，可以纳入，反之，是负激励的，不应该纳入；而没有导致元素溢出时，显然是正激励的，应该纳入。

这一模型有别于圆覆盖模型从服务节点角度搜索的特点，反而是从区域出发进行搜索。简而言之：圆覆盖模型从节点找服务对象，最近邻改进算法从服务对象出发确定节点。很显然，圆覆盖模型适用于右侧中心点分布稀疏的情形，让节点尽可能地找到自己的服务对象；而最近邻改进算法适用于左侧中心点分布密集的情形，让每个区域中心点找到自己的服务节点。

模型实际运用于左侧区域，确定节点23个。即左右两侧区域共有29个节点。

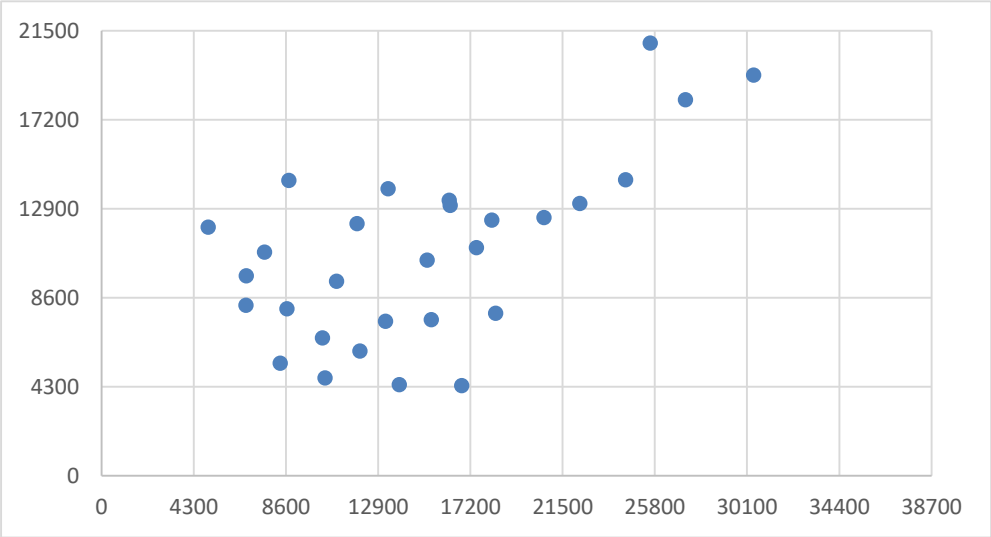


图 5-6 节点坐标分布图

5.1.3 K-means 聚类算法确定节点构成

上节对区域划分后分别运用不同模型确定了各自区域内的服务节点，这些节点都视作二级节点，接下来就是从这些二级节点中确定哪些可以作为一级节点。

将所有区域中心点的需求量在图 5-7 中画出，圆越大代表圆心点处的货物量越大，不难发现右上角稀疏，但需求量大，左下角密集，但需求量小。于是我们将右上角通过圆覆盖模型确定的二级节点划为一类，计算这 6 个点的质心，选择离质心最近的节点为一级节点，剩余 5 个二级节点归属于该一级节点。

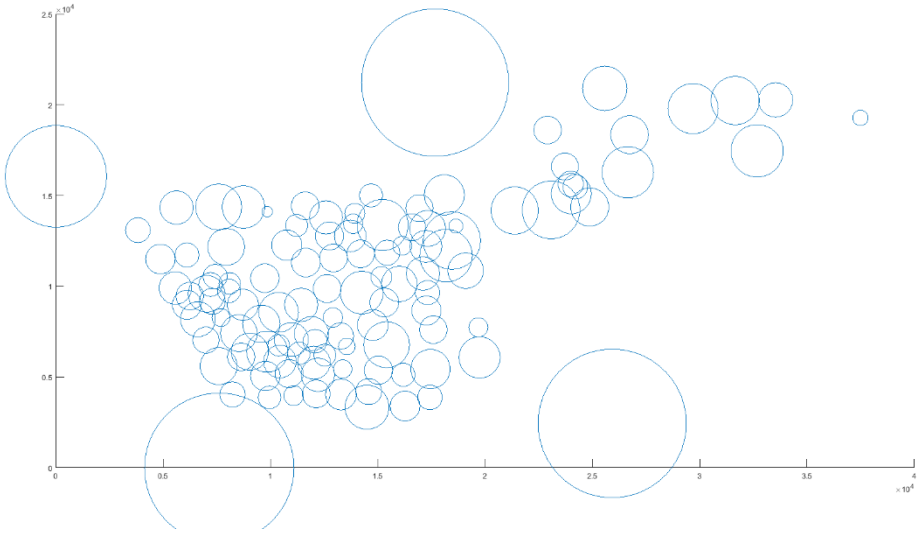


图 5-7 各区域中心点货物量散点图

对于左下角通过最近邻改进算法确定的节点，我们采用 K-means 聚类算法，将其聚类，聚类数目定为 3，因为一共有 4 个园区，而园区与一级节点连接提供货物运输服务，在已经确定右上角一个类别的情况下，左下角的聚类数目便确定为 3。

之所以采用 K-means 算法是因为作为典型的基于距离的聚类算法，其采用距离作为相似性的评价指标，与后续要降低成本的目标相契合。

鉴于 K-means 算法简单通用，本文不再赘述其算法思想和操作步骤。

需要指出的是，传统的 K-means 算法返回的类别中心为该类成员的质心，并

非一定是某一个类元素，因此，我们确定离类别中心最近的节点为一级节点。同时兼顾转运率尽量低的要求，通过编程实现，最终确定左下 23 个节点的三个类别。至此得出 4 个一级节点和 25 个二级节点，各节点的具体信息如表 5-1 所示。

表 5-1 节点信息表

类别	节点	横坐标 x（米）	纵坐标 y（米）	节点总货物量（吨）	覆盖区域	一级节点网络内总货运量（吨）
1	一级节点	143810.3	152077.3	2661.422571	800,806,809,811,818	20199.57181
	二级节点	140096.6	151487.14	2623.693191	798,813,814,810	
		142070.8	152714.59	2818.745643	815,817,819	
		142193.2	150782.56	2643.735046	793,797,801,802,807	
		145003.1	153520.63	2715.30027	816,821,828,820,824	
		145643.4	150461.63	2445.281727	791,794,796,803,804,795	
		147148	153598.53	1967.635939	822,831	
		148561.1	150408.53	2323.757424	792,799,805,808	
2	一级节点	149241.4	157084.75	2925.55523	838,839	19042.11485
	二级节点	145120.5	159927.13	2879.140457	864,865,881,878,875,883	
		146940.2	156479.93	2672.803473	835,836,837,871	
		147855.8	159877.45	2947.979761	880,882	
		148028.8	159120.91	2264.805362	870,873,876,884	
		149957	158411.43	2656.653193	840,874,872,879,877	
		150148.4	153899.61	2695.177373	812,825,826,829,832	
3	一级节点	156196.4	160363.04	2582.165937	889,891,893	15309.04963
	二级节点	152388.5	158532.49	3576.821117	886,890	
		154063.3	159209.34	2907.768377	885,887,888	
		157354.7	166956.66	1091.924754	898	
		158996.5	164226.62	2180.311785	894,900	
		162172.3	165415.23	2970.057658	892,899,897	
4	一级节点	139366.3	156856.71	2544.19032	849,851,853,859,857	21318.23648
	二级节点	136739.7	158062.48	2413.197338	852,858,862,868	
		138501.4	154299.41	2705.960376	841,842,844,845	
		138518.2	155710.16	2602.052554	846,848,850	
		140411.3	154119.13	2953.927187	823,827,843,847	
		140503.2	160321.86	2618.220482	866,867,869	
		142719.8	155455.14	2929.241719	830,833,854,834	
		143673	158241.08	2551.4465	855,856,861,860,863	

我们用四种不同的颜色：红色、黑色、绿色、蓝色，与其最近的物流园区编号一一对应，分别表示 1，2，3，4 类别，为一级节点的覆盖区域。在地图上标注出节点的服务范围，其中空白的区域是交通拥堵指数小于 4，表示能够满足交通畅通的要求，不需要从地下运输的区域。具体的节点划分图如下图所示：

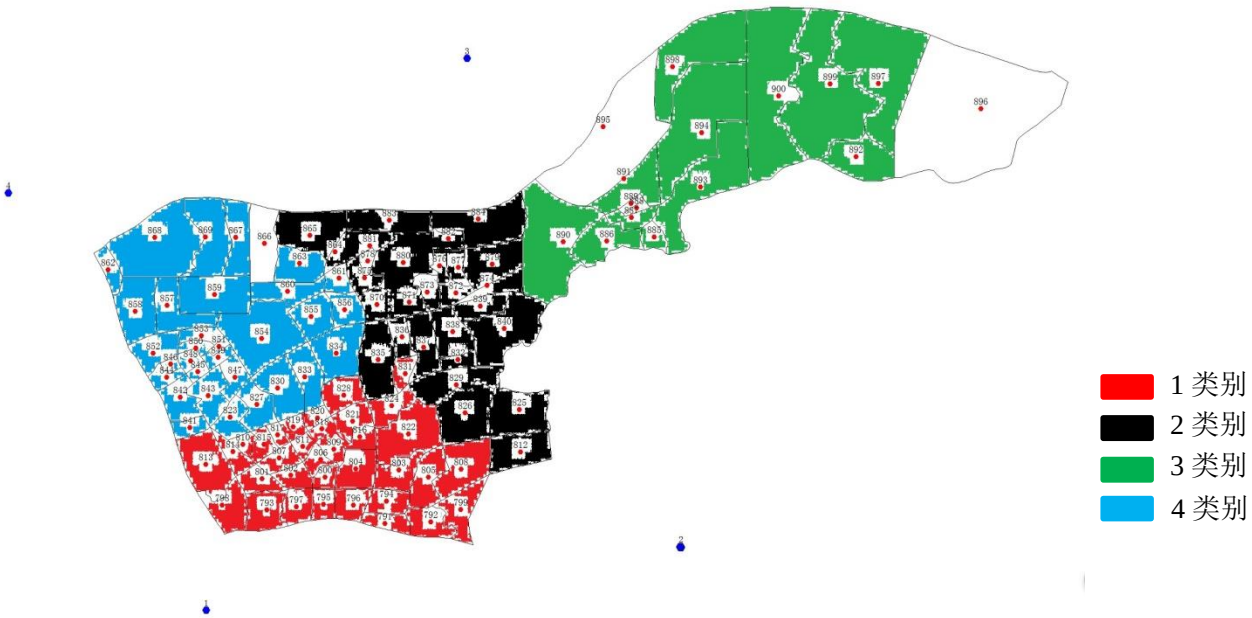


图 5-8 节点区域分布图

根据题目描述，从物流园区经由最近的一级节点转运至其他所有一级节点的货物量占该物流园区总出货量的百分比，称为该一级节点的转运率（φ），在我们构建的物流系统中，每个物流园区只与一个一级节点直接连通，因此每个物流园区的总出货量即为与之相连通的一级节点的总进货量，而一级节点转运至其他所有一级节点的货物量与总进货量的差额则为该一级节点的货物留存量。于是我们改进公式得到一级节点转运率的计算公式$\varphi = 1 - \frac{\text{一级节点留存量}}{\text{一级节点进货量}}$。

经过计算我们得到一级节点的转运率如下表：

表 5-2 一级节点转运率表

节点编号	一级节点留存量（吨）	一级节点进货量（吨）	转运率
1	2973.9381	7246.7234	58.96%
2	2083.5523	6535.8855	68.12%
3	2261.7347	6275.5265	63.96%
4	1078.5774	3138.3248	65.63%

5.2 问题二的模型建立与求解

5.2.1 数据的预处理

为了方便构建物流网络以及计算各通道的实际流量，我们需要计算整个区域全天通过地下运输的 OD 流量矩阵。

主要通过以下四个步骤。

Step1：仅选取 OD 流量矩阵中各区域的出货量，将除 4 个园区外每个横轴的出货量与该区域的比例系数 k_i 相乘，得到一个新的矩阵，该矩阵中的数值用 Q_{ij} 表示，i 代表横轴的区域编号，j 代表纵轴的区域编号，具体的编号规则如下表。每个横轴的数值为各区域实际需要通过地下运输的出货量，每个纵轴则是各区域实际需要通过地下运输的进货量；

表 5-3 区域编号对照表

区域	791	792	793	…	900	1	2	3	4
编号	0	1	2	…	109	110	111	112	113

Step2：将刚刚得出的每个纵轴的进货量进行加总，得到除 4 个物流园区外所有区域对各个区域的进货量的总和$\sum_{j=0}^{110} Q_{ij}$，并计算原 OD 流量矩阵中各纵轴的的进货量总和，分别与比例系数 k_i 相乘得到各纵轴需要通过地下运输的实际进货量的总和 Q_i，则$\sum_{j=0}^{110} Q_{ij}$与 Q_i之间的差额ΔQ_{pi}即为各区域收到 4 个物流园区对其的通过地下运输的进货量之和；

Step3：将纵轴各区域的Δ Qpi 按照 4 个物流园区总进货量的权重进行分摊，即可得出每个区域 4 个物流园区分别需要运送的进货量$Q_{ij} = \Delta\ Q_{pi} \times \frac{Q_{ij0}}{\sum_{j=110}^{11} Q_{ij}}$；

Step4：纵轴所有数值相加可以得到各区域所有的进货量的总和$\sum_{j=0}^{114} \mathrm{Q_{ij}}$，通过对比$\sum_{j=0}^{114} Q_{ij}$与 Q_i是否一致验证计算是否准确，从而得到一个完整的地下运输 OD 流量矩阵。

但是，通过计算，866、895、896 三个区域的$\Delta Q_{pi}<0$，即为 4 个物流园区对这三个区域的货运量为负值，显然，货运量不能为负，因此，该计算方式出现错误，需要对其进行优化改进。

我们分析发现，在问题一中，由于 866、895、896 三个区域的交通拥堵指数 k_i 均小于 4，我们认为这三个区域的所有进、出货物均可通过地面运输，即通过地下运输的货运量为 0，而在 Step1 的计算过程中，由于每个区域对其他所有区域均有出货量，因此根据 Step1 计算得到的地下运输 OD 流量矩阵中，866、895、896 三个区域的纵轴的进货量$Q_{ij} = Q_j \times k_i$，得出的 $Q_{ij}>0$，因此$\sum_{j=0}^{110} Q_{ij} > 0$，而各纵轴需要通过地下运输的实际进货量的总和$Q_i = 0$，从而两者的差额Δ $\mathrm{Q}_{pi} = Q_i - \sum_{j=0}^{110} Q_{ij} = 0 - \sum_{j=0}^{110} Q_{ij} < 0$。

经过分析，我们决定对原有的矩阵计算模型进行优化。

首先，由于 866、895、896 三个区域的纵轴数值产生影响，我们将经计算得出的三个区域的进货量 Q_{ij}调整为 0，保证纵轴出货量总和$\sum_{j=0}^{110} Q_{ij} = 0$，但同时发现在调整后，由于减少了纵轴数值，导致横轴的出货量相应减少，根据横轴数值计算得到的出货量总和$\sum_{i=0}^{114} Q_{ij}$少于需要转入地下运输的总出货量，将会导致大部分区域的交通拥堵指数$\mathrm{TSI}_i > 4$，则不能满足交通基本畅通的目标。

为了保证横轴出货量总和$\sum_{i=0}^{114} Q_{ij}$不能减少，我们考虑对 Step1 中的比例系数进行修正，将每个横轴的出货量总和与扣除三个区域的出货量后得到的余额之间的比值对比例系数进行修正，对于 866、895、896 这三个区域，每个区域对其

的出货量分别为 Q_{75j}，Q_{104j}，Q_{105j}，得到修正后的系数 $k_i' = k_i \times \frac{\sum_{i=0}^{114} Q_{ij}}{\sum_{i=0}^{114} Q_{ij} - Q_{75j} - Q_{101j} - Q_{105j}}$。

根据修正系数 k_i进行后续的三个步骤，得到新的地下运输 OD 流量矩阵（见附录）。

5.2.2 最优化模型

在网络节点已知的情况下，确定网络的拓扑结构即要确定节点间的边。给出目标函数：

$$\min \quad TRC + DC$$

其中：

$$TRC = TRC_{\text{园区到一级节点}} + TRC_{\text{一级节点之间}} + TRC_{\text{一级节点到二级节点}} + TRC_{\text{二级节点之间}}$$

更具体地：

$TRC_{\text{园区到一级节点}} = v_{z_i a_j} * l_{z_i a_j} * 1$，其中$z_i \in Z$，$a_j \in A$

$TRC_{\text{一级节点之间}} = v_{a_i a_j} * l_{a_i a_j} * 1$，其中$a_i, a_j \in A$，且$i \neq j$

$TRC_{\text{一级节点到二级节点}} = v_{a_i a_{ij}} * l_{a_i a_{ij}} * 1$，其中$a_i \in A$，$a_{ij} \in A'$，且$j \leq n_i$

$TRC_{\text{二级节点之间}} = v_{a_{ip} a_{iq}} * l_{a_{ip} a_{iq}} * 1$，其中$a_{ip}, a_{iq} \in A'$，且$p \leq n_i \wedge p \leq n_i$

同时：

$DC = CC * 1\%/360$，其中$CC = NC + GC$

$NC = 1.5 * m + 1 * \sum_{i=1}^{m} n_i$，分别为一级节点建设成本和二级节点建设成本

$GC = GC_{\text{园区到一级节点}} + GC_{\text{一级节点之间}} + GC_{\text{一级节点到二级节点}} + GC_{\text{二级节点之间}}$，

其中：

$GC_{\text{园区到一级节点}} = 5 * l_{z_i a_j} + 4.5 * l_{z_r a_s}$，其中$z_i, z_r \in Z$，$a_j, a_s \in A$，且$i, j, p, q$满足

$v_{z_i a_j} \geq 7200, v_{z_r a_s} < 7200$

$GC_{\text{一级节点之间}} = 3.5 * l_{a_i a_j} + 3 * l_{a_p a_q}$，其中$a_i, a_j, a_p, a_q \in A$，$i \neq j$且 i, j, p, q 满足

$v_{a_i a_j} \geq 3600$，$v_{a_p a_q} < 3600$

$GC_{\text{一级节点到二级节点}} = 3.5 * l_{a_i a_{ij}} + 3 * l_{a_p a_{pq}}$，其中$a_i, a_p \in A$，$a_{ij}, a_{pq} \in A'$，$j \leq n_i$，

$q \leq n_p$，且i, j, p, q满足$v_{a_i a_{ij}} \geq 3600$，$v_{a_p a_{pq}} < 3600$

$GC_{\text{二级节点之间}} = 3 * l_{a_{ip} a_{iq}}$，其中$a_{ip}, a_{iq} \in A'$，且$p \leq n_i \wedge p \leq n_i$

形式化地表示上述步骤为求解如下最优化问题：

$$\min \quad v_{z_i a_j} * l_{z_i a_j} * 1 + v_{a_i a_j} * l_{a_i a_j} * 1 + v_{a_i a_{ij}} * l_{a_i a_{ij}} * 1 + v_{a_{ip} a_{iq}} * l_{a_{ip} a_{iq}} * 1$$
$$+ \left(1.5 * m + 1 * \sum_{i=1}^{m} n_i + 5 * l_{z_i a_j} + 4.5 * l_{z_r a_s} + 3.5 * l_{a_i a_j} + 3\right.$$
$$\left. * l_{a_p a_q} + 3.5 * l_{a_i a_{ij}} + 3 * l_{a_p a_{pq}} + 3 * l_{a_{ip} a_{iq}}\right) * 1\%/360$$

$s.t.$ ：

1） $z_i, z_r \in Z$

2） $a_i, a_j, a_p, a_q \in A$

3） $a_{ij}, a_{ip}, a_{iq}, a_{pq} \in A'$

4） $i \neq j$， 且$j, p \leq n_i$， $q \leq n_p$

5） $v_{z_i a_j} \geq 7200,$

6） $v_{z_r a_s} < 7200$

7） $v_{a_i a_j}, v_{a_i a_{ij}} \geq 3600$

8） $v_{a_p a_q}, v_{a_p a_{pq}} < 3600$

通过求解上述最优化模型，得出最优解的边，并得到此时的最小成本。就这些最优解下文给出详细描述。

5.2.3 网络的构成

对上述模型加以实现，得出园区与一级节点、一级节点之间的通道以及一级节点与二级节点、二级节点之间的通道形成的拓扑结构。

我们将园区与一级节点、一级节点之间的通道形成的网络称为主网，一级节点与二级节点、二级节点之间的通道形成的网络称为子网。主网通道的拓扑结构与流量见图 5-9。

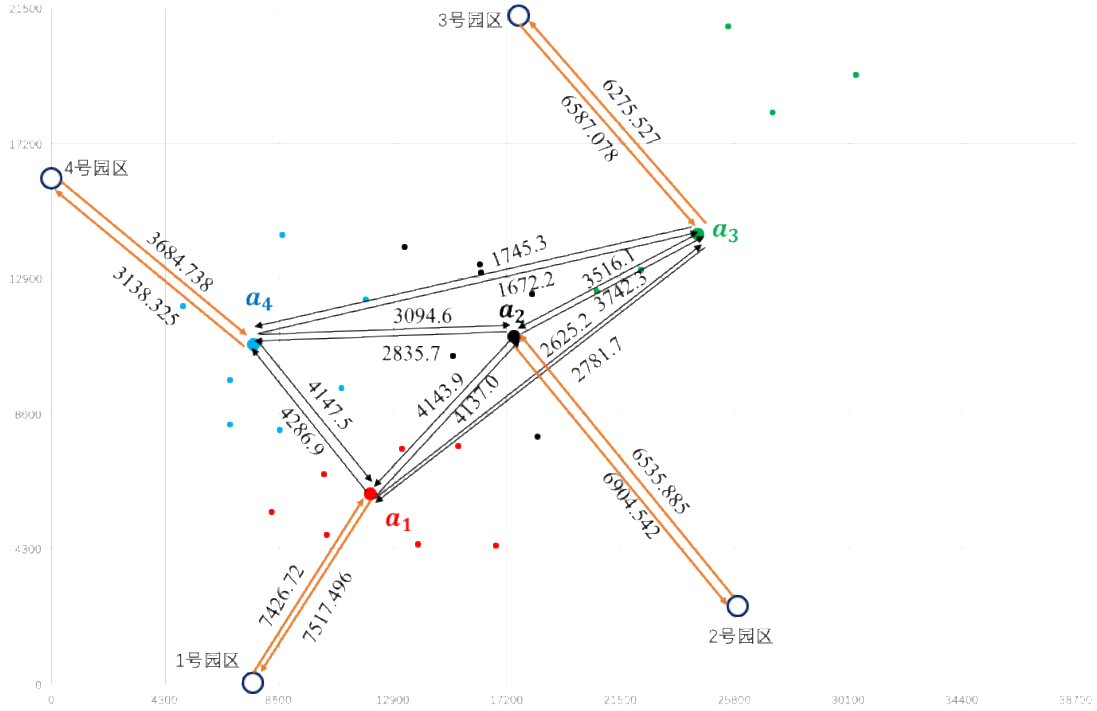


图 5-9 主网通道的位置与流量（单位：吨）

由现状 OD 数据表作主网通道实际流量表（表中数据表示横轴对纵轴的发货量，下同），见表 5-4。

表 5-4 主网通道实际流量表（单位：吨）

节点	z_1	z_2	z_3	z_4	a_1	a_2	a_3	a_4
z_1	0	0	0	0	7426.72	0	0	0
z_2	0	0	0	0	0	6535.885	0	0
z_3	0	0	0	0	0	0	6587.078	0
z_4	0	0	0	0	0	0	0	3684.738
a_1	7517.496	0	0	0	0	4137	2625.2	4286.9
a_2	0	6904.542	0	0	4143.9	0	3742.3	2835.7
a_3	0	0	6275.527	0	2781.7	3516.1	0	1745.3
a_4	0	0	0	3138.325	4147.5	3094.6	1672.2	0

子网通道的拓扑结构见图 5-10。

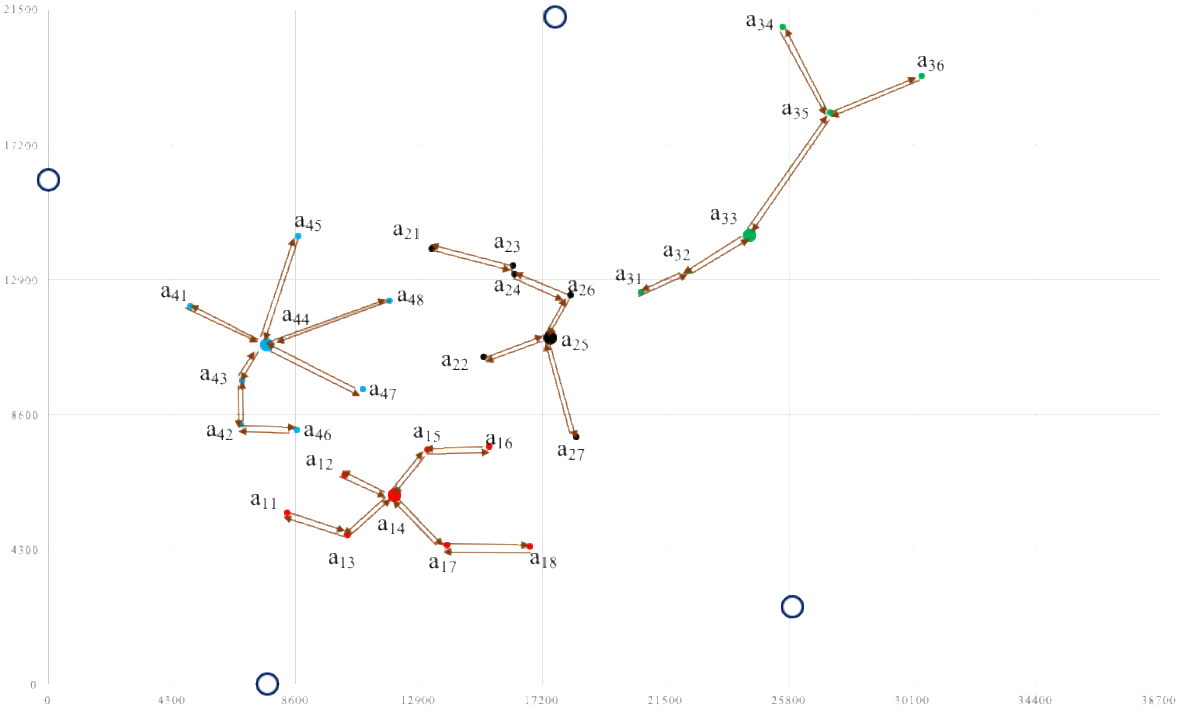


图 5-9 子网通道的拓扑结构

由现状 OD 数据表作子网 1-4 通道实际流量表，见表 5-5 至 5-8。

表 5-5 子网 1 通道实际流量表（单位：吨）

节点	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	a_{16}	a_{17}	a_{18}
a_{11}	0	1850.858	0	0	0	0	0	0
a_{12}	1325.179	0	0	3174.177	0	0	0	0
a_{13}	0	0	0	1747.809	0	0	0	0
a_{14}	0	3609.503	1347.09	0	2793.206	2590.451	0	0
a_{15}	0	0	0	2925.544	0	0	1016.861	0
a_{16}	0	0	0	2742.766	0	0	0	1392.848
a_{17}	0	0		0	1068.284	0	0	0
a_{18}	0	0	0	0	0	1430.801	0	0

表 5-6 子网 2 通道实际流量表（单位：吨）

节点	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}
a_{21}	0	0	2976.3	0	0	0	0
a_{22}	0	0	0	0	1290.475	0	0
a_{23}	2654.31	0	0	4585.79	0	0	0
a_{24}	0	0	2856.82	0	0	3492.51	0
a_{25}	0	2399.581	0	0	0	3528.135	1537.143
a_{26}	0	0	0	3670.19	6624.856	0	0
a_{27}		0	0	0	1529.508	0	0

表 5-7 子网 3 通道实际流量表（单位：吨）

节点	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}
a_{31}	0	2114.29	0	0	0	0
a_{32}	1653.97	0	2904.81	0	0	0
a_{33}	0	2499.89	0	0	**3644.856**	0
a_{34}	0	0	0	0	959.06	0
a_{35}	0	0	**3614.501**	678.98	0	1909.67
a_{36}	0	0	0	0	1958.58	0

表 5-8 子网 4 通道实际流量表（单位：吨）

节点	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{41}	0	0	0	1275.23	0	0	0	0
a_{42}	0	0	2688.6	0	0	1216.76	0	0
a_{43}	0	2502.02	0	**4200.59**	0	0	0	0
a_{44}	1825.63	0	**3676.09**	0	1950.08	0	2584.25	1645.23
a_{45}	0	0	0	2252.37	0	0	0	0
a_{46}	0	1238.5	0	0	0	0	0	0
a_{47}	0	0	0	1615.07	0	0	0	0
a_{48}	0	0	0	2006.74	0	0	0	0

各节点实际货运量见表 5-9

表 5-9 各节点实际货运量（单位：吨）

子网类别	节点类型	节点编号	节点实际货运量
1	一级节点	a_1/a_{14}	20981.808
	二级节点	a_{11}	3176.037
		a_{12}	3146.166
		a_{13}	9959.717
		a_{15}	7803.89
		a_{16}	8156.866
		a_{17}	2085.145
		a_{18}	2823.649
2	一级节点	a_2/a_{25}	19781.698
	二级节点	a_{21}	5630.4
		a_{22}	3690.056
		a_{23}	13074.01
		a_{24}	16570.31
		a_{26}	22151.691
		a_{27}	3066.651
3	一级节点	a_3/a_{33}	12664.057
	二级节点	a_{31}	3768.28
		a_{32}	9172.98
		a_{34}	1638.04
		a_{35}	12765.647
		a_{36}	3868.25
4	一级节点	a_4/a_{44}	22833.16
	二级节点	a_{41}	3100.86
		a_{42}	7645.88
		a_{43}	12869.3
		a_{45}	4202.45
		a_{46}	2455.26
		a_{47}	4199.32
		a_{48}	3651.97

5.2.4 每日总成本的计算

每天的总成本由货物的运输成本和地下物流隧道与节点的折旧费用构成。

（1）运输成本

每吨货每公里的平均运输成本始终相等，约为 1 元/吨·公里，根据主网及各子网的实际流量表，计算得出各区域的运输成本如下表：

表 5-10 运输成本表（单位：元）

区域	运输成本
主网	831078.60
子网 1	62816.52
子网 2	70944.19
子网 3	72887.92
子网 4	87198.29
总计	1124925.52

（2）通道建设成本

题中给出车辆参数，两种型号的车辆设计载重分别为 10 吨和 5 吨，一班车一般由四只八节车辆构成，每个节点每 12 分钟可发车一般，每小时最多发车 5 班，每天可以运营 18 小时。为使每班车达到最大承载量，我们使用载重为 10 吨的车辆，每班车由八节车辆构成。

根据题中所述，一级节点与物流园区相连且采用 10 吨的大型车辆地下运输，除园区至一级节点的地下通道外其他地下通道均采用 5 吨的地下运输车辆。

因此，在园区与一级节点间，如果建设双向双轨（双洞）的隧道，则每天单向最大流量为$Q = 5 \times 18 \times 10 \times 8 = 7200$（吨），如果建设双向四轨（双洞）的隧道，则每天单向最大流量为$Q = 5 \times 18 \times 10 \times 8 \times 2 = 14400$（吨）；

对于其他通道而言，如果建设双向双轨（双洞）的隧道，则每天单向最大流量为$Q = 5 \times 18 \times 5 \times 8 = 3600$（吨），如果建设双向四轨（双洞）的隧道，则每天单向最大流量为$Q = 5 \times 18 \times 10 \times 8 \times 2 = 7200$（吨）。

据题意，每两个相邻节点间地下物流通道双向尺寸一致，以单项流量较大的为设计原则，因此，我们将主网及子网实际流量表超过单向最大流量的通道数值在表中进行标记，得到需要建设双向四轨（双洞）的通道如下：

园区与一级节点的通道：$e_{z_1a_1}$

一级节点之间的通道：$e_{a_1a_2}$，$e_{a_2a_3}$，$e_{a_1a_4}$

一级节点与二级节点之间及二级节点之间的通道：

子网 1：$e_{a_{14}a_{12}}$

子网 2：$e_{a_{23}a_{24}}$，$e_{a_{24}a_{26}}$，$e_{a_{25}a_{26}}$

子网 3：$e_{a_{33}a_{35}}$

子网 4：$e_{a_{43}a_{44}}$

地下物流隧道的建设成本为：双向四轨（双洞）（10 吨）造价 5 亿元/公里，双向双轨（双洞）（10 吨）造价 4 亿元/公里，双向四轨（双洞）（5 吨）造价 3.5

亿元/公里，双向双轨（双洞）（5 吨）造价 3 亿元/公里，根据计算公式，可以分别得到以下各通道间的总建设成本如下表：

表 5-11 通道建设成本表（单位：亿元）

通道	轨道类型	建设成本	总计
园区与一级节点之间通道	双向双轨	123.82	161.23
	双向四轨	37.41	
一级节点之间通道	双向双轨	145.49	197.21
	双向四轨	51.72	
四个子网内部通道	双向双轨	164.24	207.60
	双向四轨	43.36	
总计		566.04	

（3）节点建设成本

根据问题一确认的节点设置，我们建造一级节点 4 个，二级节点 25 个，共 29 个节点，一级节点的建设成本约为 1.5 亿元/个，二级节点的建设成本约为 1 亿元/个，因此节点建设成本为 31 亿元。

（4）折旧费用

根据通道与节点的建设成本可以得到该物流网络的总建设成本为$\mathrm{TRC} = 566.04 + 31 = 597.04$（亿元）。

通道与节点的设计年限 100 年，年综合折旧率均为 1%。我们假定通道与节点每天的折旧率根据会计准则按照每年 360 天进行计算，则每天的折旧费用$\mathrm{DC} = \frac{\mathrm{TRC}\times 1\%}{360} = \frac{597.04\times 1\%}{360} = 0.01658444(\text{亿元}) = 165.84$（万元）。

综合上述计算，得到该物流网络每天的总成本$\mathrm{TC} = \mathrm{CC} + \mathrm{DC} = 112.49 + 165.84 = 278.33$（万元）。

5.3 问题三的模型建立

5.3.1 实际运行时轨道状况的分析

根据问题二中的分析，为最大化运行，假设轨道的一班车均由最大车辆数 8 辆车构成，并且以最大频率发车即每小时发车 8 班，在满足地下节点及通道内无货物滞留的条件下，5 吨车双向双轨（双洞）日最大货运量为 3600 吨，双向四轨（双洞）日最大货运量为 7200 吨；10 吨车双向双轨（双洞）日最大货运量为 7200 吨，双向四轨（双洞）日最大货运量为 14400 吨。

依题意，园区与一级节点的轨道可以用 10 吨车，双轨可满足的最大货运量为 3600 吨，故通过分析主网的流量矩阵，可以确定轨道 $e_{z_1a_1}$需采用 10 吨的双向四轨（双洞）轨道，其货运能力达到 14400 吨/日。同理分析主网的流量矩阵和子网的流量矩阵，可以确定$e_{a_1a_2}, e_{a_2a_3}, e_{a_1a_4}, e_{a_{14}a_{12}}, e_{a_{23}a_{24}}, e_{a_{24}a_{26}}, e_{a_{25}a_{26}}, e_{a_{33}a_{35}}, e_{a_{43}a_{44}}$这些轨道均因流量大于临界值，需要采用双向四轨（双洞）轨道。

进一步地可以得出主网和四个子网的轨道最大承载规模矩阵。如表所示：

表 5-12 网轨道最大承载量表

节点	z_1	z_2	z_3	z_4	a_1	a_2	a_3	a_4
z_1	0	0	0	0	14400	0	0	0
z_2	0	0	0	0	0	7200	0	0
z_3	0	0	0	0	0	0	7200	0
z_4	0	0	0	0	0	0	0	7200
a_1	14400	0	0	0	0	7200	7200	7200
a_2	0	7200	0	0	7200	0	7200	7200
a_3	0	0	7200	0	7200	7200	0	7200
a_4	0	0	0	7200	7200	7200	7200	0

表 5-13 子网 1 轨道最大承载量表

节点	$\boldsymbol{a_{11}}$	$\boldsymbol{a_{12}}$	$\boldsymbol{a_{13}}$	$\boldsymbol{a_{14}}$	$\boldsymbol{a_{15}}$	$\boldsymbol{a_{16}}$	$\boldsymbol{a_{17}}$	$\boldsymbol{a_{18}}$
$\boldsymbol{a_{11}}$	0	3600	0	0	0	0	0	0
$\boldsymbol{a_{12}}$	3600	0	0	0	0	0	0	0
$\boldsymbol{a_{13}}$	0	0	0	0	0	0	0	0
$\boldsymbol{a_{14}}$	0	7200	3600	0	3600	3600	0	0
$\boldsymbol{a_{15}}$	0	0	0	3600	0	0	3600	0
$\boldsymbol{a_{16}}$	0	0	0	3600	0	0	0	3600
$\boldsymbol{a_{17}}$	0	0	0	0	3600	0	0	0
$\boldsymbol{a_{18}}$	0	0	0	0	0	3600	0	0

表 5-14 子网 2 主网轨道最大承载量表

节点	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}
a_{21}	0	0	3600	0	0	0	0
a_{22}	0	0	0	0	3600	0	0
a_{23}	3600	0	0	7200	0	0	0
a_{24}	0	0	7200	0	0	3600	0
a_{25}	0	3600	0	0	0	3600	3600
a_{26}	0	0	0	0	7200	0	0
a_{27}	0	0	0	0	3600	0	0

表 5-15 子网 3 轨道最大承载量表

节点	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}
a_{31}	0	3600	0	0	0	0
a_{32}	3600	0	3600	0	0	0
a_{33}	0	3600	0	0	7200	0
a_{34}	0	0	0	0	3600	0
a_{35}	0	0	7200	3600	3600	3600
a_{36}	0	0	0	0	3600	0

表 5-16 子网 4 轨道最大承载量表

节点	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{41}	0	0	0	3600	0	0	0	0
a_{42}	0	0	3600	0	0	3600	0	0
a_{43}	0	3600	0	7200	0	0	0	0
a_{44}	3600	0	7200	0	3600	0	3600	3600
a_{45}	0	0	0	3600	0	0	0	0
a_{46}	0	3600	0	0	0	0	0	0
a_{47}	0	0	0	3600	0	0	0	0
a_{48}	0	0	0	3600	0	0	0	0

进而得出主网与子网的轨道利用率如表所示：

表 5-17 主网轨道利用率

节点	z_1	z_2	z_3	z_4	a_1	a_2	a_3	a_4
z_1	0	0	0	0	0.52	0	0	0
z_2	0	0	0	0	0	0.91	0	0
z_3	0	0	0	0	0	0	0.91	0
z_4	0	0	0	0	0	0	0	0.51
a_1	0.52	0	0	0	0	0.57	0.36	0.60
a_2	0	0.96	0	0	0.58	0	0.52	0.39
a_3	0	0	0.87	0	0.39	0.49	0	0.24
a_4	0	0	0	0.44	0.58	0.43	0.23	0

表 5-18 子网 1 轨道利用率

节点	$\boldsymbol{a_{11}}$	$\boldsymbol{a_{12}}$	$\boldsymbol{a_{13}}$	$\boldsymbol{a_{14}}$	$\boldsymbol{a_{15}}$	$\boldsymbol{a_{16}}$	$\boldsymbol{a_{17}}$	$\boldsymbol{a_{18}}$
$\boldsymbol{a_{11}}$	0	0	0	0	0.52	0	0	0
$\boldsymbol{a_{12}}$	0	0	0	0	0	0.91	0	0
$\boldsymbol{a_{13}}$	0	0	0	0	0	0	0.91	0
$\boldsymbol{a_{14}}$	0	0	0	0	0	0	0	0.51
$\boldsymbol{a_{15}}$	0.52	0	0	0	0	0.57	0.36	0.60
$\boldsymbol{a_{16}}$	0	0.96	0	0	0.58	0	0.52	0.39
$\boldsymbol{a_{17}}$	0	0	0.87	0	0.39	0.49	0	0.24
$\boldsymbol{a_{18}}$	0	0	0	0.44	0.58	0.43	0.23	0

表 5-19 子网 2 轨道利用率

节点	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}	a_{26}	a_{27}
a_{21}	0	0	0.83	0	0	0	0
a_{22}	0	0	0	0	0.36	0	0
a_{23}	0.74	0	0	0.64	0	0	0
a_{24}	0	0	0.40	0	0	0.97	0
a_{25}	0	0.67	0	0	0	0.98	0.43
a_{26}	0	0	0	0	0.92	0	0
a_{27}	0	0	0	0	0.42	0	0

表 5-20 子网 3 轨道利用率

节点	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}	a_{36}
a_{31}	0	0.59	0	0	0	0
a_{32}	0.46	0	0.81	0	0	0
a_{33}	0	0.69	0	0	0.51	0
a_{34}	0	0	0	0	0.27	0
a_{35}	0	0	0.50	0.19	0	0.53
a_{36}	0	0	0	0	0.54	0

表 5-21 子网 4 轨道利用率

节点	a_{41}	a_{42}	a_{43}	a_{44}	a_{45}	a_{46}	a_{47}	a_{48}
a_{41}	0	0	0	0.35	0	0	0	0
a_{42}	0	0	0.75	0	0	0.34	0	0
a_{43}	0	0.70	0	0.58	0	0	0	0
a_{44}	0.51	0	0.51	0	0.54	0	0.72	0.46
a_{45}	0	0	0	0.63	0	0	0	0
a_{46}	0	0.34	0	0	0	0	0	0
a_{47}	0	0	0	3600	0	0	0	0
a_{48}	0	0	0	3600	0	0	0	0

定义 5.2（节点负荷度 τ） 流经某一节点的实际货运量与该级别所有节点均值的比值与 1 的差值。其计算公式为：

$$\tau = \frac{\text{流经节点的货运总量}}{\text{与节点同级所有节点货运总量的均值}} - 1$$

不难得知，负荷度反映了节点相对于同类节点的负载情况，值为正说明该节点负荷偏多，越大负荷越重；值为负说明该节点负荷偏小，值越小负荷越轻。

依上述公式计算得所有节点的负荷度如表 5-22 所示：

表 5-22 所有节点的负荷度

子网类别	节点类型	节点编号	负荷度
1	一级节点	a14	0.1005302
	二级节点	a11	-0.534777
		a12	-0.539152
		a13	0.4588903
		a15	0.1431067
		a16	0.1948103
		a17	-0.69457
		a18	-0.586394
2	一级节点	a25	0.0375825
	二级节点	a21	-0.175264
		a22	-0.459484
		a23	0.9150691
		a24	1.4272039
		a26	2.2447595
		a27	-0.5508
3	一级节点	a33	0.1976367
	二级节点	a31	-0.448026
		a32	0.3436498
		a34	-0.760061
		a35	0.8699003
		a36	-0.433382
4	一级节点	a44	-0.335749
	二级节点	a41	-0.545789
		a42	0.1199615
		a43	0.8850833
		a45	-0.384429
		a46	-0.640356
		a47	-0.384887
		a48	-0.465063

5.3.2 网络优化部分的选择

分析上述表格，寻找到管道利用率高于 0.9 和低于 0.3 的管道如表 5-23 所示：

表 5-23 管道利用率极端值

$\sigma \geq 0.9$		$\sigma < 0.3$	
管道	利用率（双向）	管道	利用率（双向）
$e_{z_2a_2}$	0.96/0.91	$e_{a_3a_4}$	0.23/0.24
$e_{a_{25}a_{26}}$	0.97/0.98	$e_{a_{15}a_{16}}$	0.39/0.28
		$e_{a_{34}a_{35}}$	0.19/0.27

可以发现管道利用率偏高的，其两端节点的负荷度也偏高，这是符合正常情况的。我们选择优化的部分就限于这些部分，我们并不对利用率低的管道和负荷度低的节点加以调整，因为它们可以充当整个网络中抗风险的部分。

在新建节点和调整节点级别的选择上，我们参考问题二中在成本计算时的直观感受，管道的铺设成本偏大，而新增节点则必然带来管道的铺设工作。故我们选择通过调整节点级别来进行调整网络。

5.4 问题四的模型建立

根据题目中要求，希望考虑满足该市近 30 年内的交通需求，但是前三个问题都是建立在南京市仙林区的基础上，因此我们认为此处题目出现错误，应该改为满足该区域 30 年内的交通需求。交通需求每年呈 5%的增长率增加，我们假定交通需求即货运需求量每年呈 5%增长，不考虑其他因素，结合实际情况，道路容量及人口密度等也将相应增长，假定道路容量与人口密度的增长比例相同，各区域交通拥堵指数基本维持不变。为满足题中 30 年内的交通需求，我们首先计算出 30 年后的需求量与目前的货物需求量的比率$k = (1 + 5\%)^{30} = 1.05^{30} = 4.3219$，因此 30 年后该区域的货运需求量将达到目前需求量的 4.32 倍。

5.4.1 扩容的相关方案建议

问题三中设计的地下物流网络仅考虑了目前的货物流量，可以达到缓解当前交通拥堵的目标。随着货物需求量逐年增长，该物流网络将不能承载日益增加的货物流量，因此，对于我们目前建立的物流网络，我们需要对该系统进行扩容处理。

我们拟采用以下方案进行扩容：

（1）双向双轨通道改建为双向四轨通道

由于货物需求量每年呈 5%增长，30 年后的货物需求量将达到目前需求量的 4 倍以上，因此，对于通道需要进行扩容。为了最大程度地满足货物需求，所有节点之间的物流运输都选择使用载重 10 吨的车辆，每班车由八节车厢组成。

题中指出，双向四轨（双洞）（10 吨）通道的造价为 5 亿元/公里，双向双轨（双洞）（10 吨）通道的造价为 4 亿元/公里，则可以认为双向四轨（双洞）（10 吨）的通道每条轨道的造价为 1.2 亿元/公里，双向双轨（双洞）（10 吨）的通道每条轨道的造价为 2 亿元/公里，考虑节约成本及扩大运输能力，可以将所有一、二级节点与园区之间的双向双轨通道全部改建为双向四轨通道。

（2）根据需求增长速度增设一级节点

根据 30 年内的交通需求，货运量将达到目前货运量的 4.32 倍，因此仅以目前设置的一级节点数必然不能满足未来的货运需求。

我们发现，各个区域之间互相运输的货运量非常小，每个节点的实际总货运量大部分由园区向节点的进货量构成，而二级节点仅负责本区域内的货物运输，向区域外的节点运输货物需要通过一级节点转运，因此考虑增加一级节点数量。由于 30 年内货运量增长，二级节点间的货运量也相应增加，因此为满足二级节点间的货运需求，不考虑将二级节点改设为一级节点，仅通过增加一级节点数量及通道满足园区与一级节点、一级节点与一级节点间以及转运的需求。

（3）调整检修时间，延长班车运营时间

参考车辆参数，每个节点每 12 分钟可发车 1 班，每小时最多发车 5 班，每天可以运营 18 小时，剩余 6 小时为检修时间，可以认为其意义是运营时间结束后将所有车辆集中统一进行检修。根据实际情况，将原有的集中检修时间分批次，将所有车辆分批，每批次车辆检修需要固定时长，检修后的车辆继续投入运营。

具体车辆的检修时间需要更多数据来确定，但这一固定时长必然不超过原有的 6 小时检修时间，可以达到延长运营时间的目的，以保证更多时间可以进行货物运输，增加货物运输容量。

5.4.2 地下物流系统的建设时序及演进过程

根据拟采用的扩容方案，我们首先需要确定增设的一级节点的数量及位置，通过园区与一级节点间的互连，根据问题三确定的模型通过减少节点间通道长度调整一级节点的位置。通过对一、二级节点的增设及通道的调整，基本可以形成树状地下物流网络系统。

由于每年可建设通道长度大致相等，我们需要计算出园区与一级节点之间的通道长度、一级节点之间的通道长度、一级节点与二级节点之间以及二级节点之间的通道长度的比例为$\sum l_{z_i a_j}：\ \sum l_{a_i a_j}：\ \sum(l_{a_i a_{ij}} + l_{a_{ip} a_{iq}})$，根据比例确定四类通道长度的权重。

由于园区与一级节点间的货运量较大，我们在考虑建设时序时首先完成一级节点及节点间通道的建设，接着进行园区与一级节点间通道的建设，最后是一级节点与二级节点间、二级节点及节点间通道同时建设。

对求得的通道长度进行计算，由于时间限制，我们只能得到各类通道的长度间的近似比例大致为$\sum l_{z_i a_j}：\ \sum l_{a_i a_j}：\ \sum(l_{a_i a_{ij}} + l_{a_{ip} a_{iq}}) = 2:3:3$，因此我们可以得到园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道需要的建设年限的权重分别为$\frac{2}{8}$，$\frac{3}{8}$，$\frac{3}{8}$。

根据题目要求，需要分八年完成地下物流网络系统的建设，因此按照权重分配建设年限，计算出园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道需要的建设年限分别为两年，三年，三年。

按照地下物流系统线路建设的顺序，三年内完成一级节点及节点间通道的建设，第四年和第五年进行园区与一级节点间通道的建设，最后三年完成一级节点与二级节点间、二级节点及节点间通道的建设。

六、总结

随着城市货运物流需求的快速增长，构建地下物流系统对于缓解城市交通拥堵、减轻城市污染具有重要的意义。本文针对南京市仙林地区的地下物流运输问题，通过运用圆覆盖算法、基于最近邻的改进算法寻找二级节点，通过 K-means 算法确定一级节点，通过最优化模型构建出地下物流系统网络模型。该模型以缓解交通拥堵、降低物流成本为目标，较为科学合理地设计出网络构成，并综合考虑南京市未来的交通需求增长与系统抗风险能力，设计出地下物流系统的建设时序及动态演进过程。

问题一：以区域交通拥堵指数降为 4 则认为达到交通基本畅通为条件，根据其与 OD 货运流量矩阵数据成正比，计算出各区域需要转入地下物流系统运输的货运量。根据区域中心点的离散程度及区域整体形状，以 890 区域的左边线为界，将整个区域划分为左侧区域与右侧区域：在右侧区域运用圆覆盖模型，在左侧区域运用基于最近邻的改进算法，最终找出所有节点共 29 个，分别给出位置坐标；接着运用 K-means 聚类算法确定一级节点共 4 个，剩余为二级节点共 25 个；最终确定各级节点的服务区域及实际货运量，以及各一级节点的转运率：最小为 58.96%，最大为 68.12%。

问题二：在地下物流网络节点群的基础上，选择合适的地下路线以建立该区域的“地下物流系统”网络。我们先对各区域全天 OD 流量矩阵进行预处理，得到每天需要通过地下运输的 OD 流量矩阵。以总成本为目标函数，在满足运输需求的约束下最小化目标函数，通过求解最优解，可以找到最合适的边进而确定网络的拓扑结构。根据网络拓扑结构可以进一步确定各节点的实际货运量和网络中管道的实际流量，计算出该地下物流网络的建设成本为 597.04 亿元，每天的运输成本为 112.49 万元，折旧费用为 165.84 万元，最后得到每天的总成本为 278.33 万元。

问题三：在问题一和问题二已经确定整个网络结构的情况下，仿真运输过程，定义管道中实际流量与管道最大负载流量的比值为管道利用率；定义各节点实际货物流量与所有同级节点货物流量均值的比值与 1 的差为节点的负荷度。通过分析数据，发现在现有网络下，发现主网（园区-一级节点共同构成的网络层）中管道$e_{z_3a_3}$利用率偏高，管道$e_{a_3a_4}$利用率偏低；子网（一级节点下的所属网络）中管道$e_{a_{25}a_{26}}$利用率偏高，管道$e_{a_{34}a_{35}}$利用率偏低。节点方面a_{24}, a_{26}负荷度偏高，a_{43}偏低。针对利用率高管道和负荷度高的节点，在其周围适当增设服务节点，对利用率较低的管道和负荷度较低的节点不作改变，用作抗风险的备用。经过调整以后网络的运输成本有所下降，并且运输速度显著提高。

问题四：对 ULS 做好顶层设计，使其能够满足该区域近 30 年交通需求的目标。交通需求每年呈 5%的增长率增加，我们求得 30 年后的货运需求为目前货运量的 4.32 倍，为满足需求量，我们提出三项扩容方案，包括将双向双轨通道改建为双向四轨，增设一级节点以及延长班车运营时间。由于每年可建设道路长度大致相等，根据园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道长度之间的比例确定权重，按照权重分配建设年限，计算出三类通道需要的建设年限分别为 2 年，3 年，3 年。

七、模型评价及推广

7.1 模型的优缺点评价

本文根据题目要求，通过“确定物流节点位置”、“通道网络设计”、“从全局考虑进行网络改进”以及“建设时序与动态优化”四个步骤，采取了分步建模，逐级优化的策略，将复杂的模型简化到可以求解的程度，并且在对可行解的分析过程中，发现了可行解的一些特点，根据这些特点我们设计了尽量合理的可行解筛选步骤，从而得到的解虽然不是最优解，但应该是一个相对近似的最优解。

但是由于我们做了一些简化模型的假设，因此本文中建立的模型也存在以下问题。

在问题一中降低交通拥堵指数至 4 时，仅认为交通状况达到基本畅通即满足目标要求，没有考虑通道中断、货运量激增等可能导致货运量上升的意外状况，因此在该模型下，一旦发生极端状况，则交通拥堵指数会超过 4，交通状况则升级为轻度拥堵，不能满足目标要求。

界定区域货运量与面积之比（即量面比）过小的范围时，仅根据平均量面比以及各区域量面比的分布情况来确定，并没有太多科学依据，因先确定这些区域的量面比服从的概率分布情况，通过求解置信区间、显著性水平等确定过小的临界值，而题中为简化工作量，没有进行这些计算。

地下物流系统的基本特征中包括：对于货物单一、流量大且稳定、地面交通差的区域应优先建立线路。但在问题四规划各线路的建设时序时，为了方便计算，我们仅将通道的建设分为园区与一级节点之间的通道、一级节点之间的通道、一级节点与二级节点之间以及二级节点之间的通道三大类，没有对上述特殊区域进行优先考虑。

7.2 模型的推广

地下物流系统网络模型是实现城市可持续发展的物流运输体系的一个有效方式，对于减轻城市环境污染，节约城市地面土地资源，实现城市可持续发展、改善城市品位，优化城市经济结构等方面有着不可忽视的作用。

本地下物流系统网络模型不仅可以适用于南京市仙林区，因其为非人口高度密集地区，近似认为区域的交通指数与区域货运总量直接相关，因此同样可以适用于南京市其他人口密度较小且有较大的城区经济发展空间的区划，如江宁区、浦口区，以及六合区、高淳区、溧水区等。

除了物流的运输之外，地下运输系统还可以应用到城市生活垃圾的收集、能源化工行业的资源运输（比如工业矿石的运输）。

参考文献

[1] Berner F, Hermes M, Lange S, et al. Lean Construction in the logistics of underground and road construction Optimization of construction in constricted conditions[J]. BAUINGENIEUR. 2016, 91: 166-171.
[2] 李鹏，朱合华，王璇，等. 地下物流系统对城市可持续发展的作用探讨[J]. 地下空间与工程学报. 2007(01): 1-4.
[3] Vernimmen B, Dullaert W, Geens E, et al. Underground Logistics Systems: A Way to Cope with Growing Internal Container Traffic in the Port of Antwerp[J]. Transportation Planning and Technology. 2007, 30(4): 391-416.
[4] 胡建军，唐常杰，李川，等. 基于最近邻优先的高效聚类算法[J]. 四川大学学报(工程科学版). 2004(06): 93-99.
[5] 储岳中，徐波. 动态最近邻聚类算法的优化研究[J]. 计算机工程与设计. 2011(05): 1687-1690.
[6] 张建萍，刘希玉. 基于聚类分析的 K-means 算法研究及应用[J]. 计算机应用研究. 2007(05): 166-168.
[7] 杨善林，李永森，胡笑旋，等. K-MEANS 算法中的 K 值优化问题研究[J]. 系统工程理论与实践. 2006(02): 97-101.
[8] 肖晓伟，肖迪，林锦国，等. 多目标优化问题的研究概述[J]. 计算机应用研究. 2011(03): 805-808.

附录

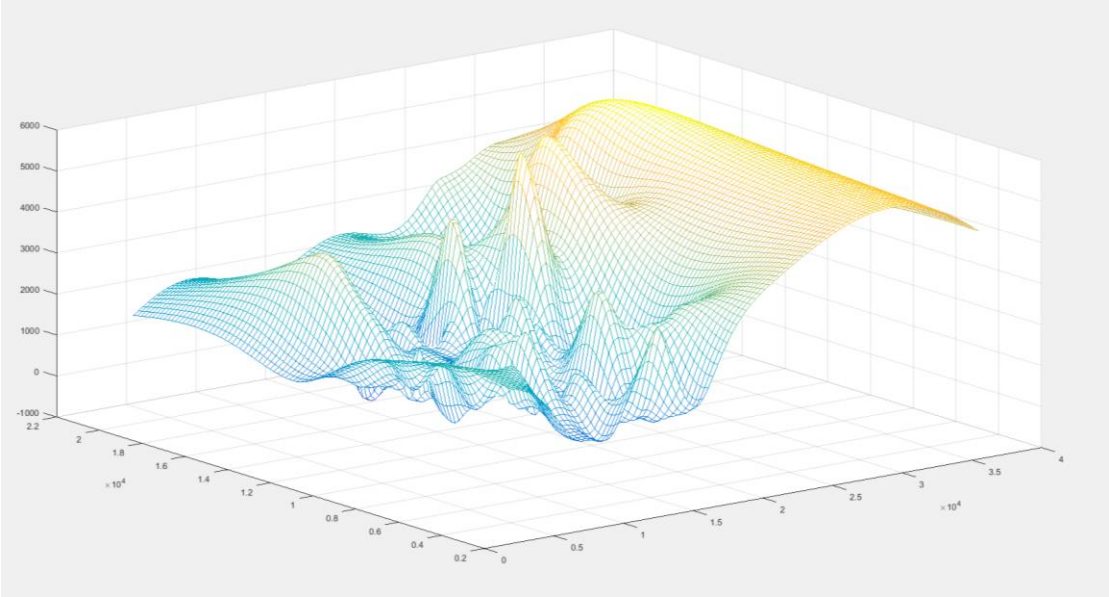
附件 1 坐标调整前后对照表

ZONE	原坐标		调整后坐标	
	中心点 X（米）	中心点 Y（米）	中心点 x（米）	中心点 y（米）
1	139382.3437	146047.0951	7617.6307	0.0000
2	157688.4185	148489.0310	25923.7055	2441.9359
3	149441.1622	167278.6394	17676.4492	21231.5443
4	131764.7130	162103.0925	0.0000	16055.9974
791	146277.0000	149377.0500	14512.2870	3329.9549
792	148037.4200	149443.9000	16272.7070	3396.8049
793	141732.2300	149908.0200	9967.5170	3860.9249
794	146343.9800	150237.0900	14579.2670	4189.9949
795	143903.2700	150127.8400	12138.5570	4080.7449
796	145062.6800	150092.5500	13297.9670	4045.4549
797	142863.2700	150030.9400	11098.5570	3983.8449
798	139998.8100	150082.7200	8234.0970	4035.6249
799	149198.4600	149918.9200	17433.7470	3871.8249
800	143976.3000	151159.1600	12211.5870	5112.0649
801	141546.8700	151107.9000	9782.1570	5060.8049
802	142646.6000	151242.9600	10881.8870	5195.8649
803	146813.3800	151435.8200	15048.6670	5388.7249
804	145146.7700	151489.5600	13382.0570	5442.4649
805	147951.6800	151173.0700	16186.9670	5125.9749
806	143800.1000	151838.3900	12035.3870	5791.2949
807	142199.9400	151897.7700	10435.2270	5850.6749
808	149214.8100	151472.2300	17450.0970	5425.1349
809	144302.0500	152239.5000	12537.3370	6192.4049
810	140808.2000	152432.4700	9043.4870	6385.3749
811	143106.6700	152356.1300	11341.9570	6309.0349
812	151500.7200	152128.3200	19736.0070	6081.2249
813	139366.7400	151652.5900	7602.0270	5605.4949
814	140416.5700	152146.0800	8651.8570	6098.9849
815	141612.0200	152424.7800	9847.3070	6377.6849
816	145313.2100	152711.9500	13548.4970	6664.8549
817	142147.4400	152788.0900	10382.7270	6740.9949
818	143839.8000	153020.8300	12075.0870	6973.7349
819	142733.3500	153099.5900	10968.6370	7052.4949
820	143678.0600	153444.6400	11913.3470	7397.5449
821	145035.9900	153315.7000	13271.2770	7268.6049
822	147188.7900	152823.1500	15424.0770	6776.0549
823	140309.1500	153473.7800	8544.4370	7426.6849
824	146543.6800	153913.7800	14778.9670	7866.6849

825	151449.4400	153764.2500	19684.7270	7717.1549
826	149366.8900	153648.0500	17602.1770	7600.9549
827	141339.4500	153959.7200	9574.7370	7912.6249
828	144690.9800	154316.9900	12926.2670	8269.8949
829	149026.5600	154716.6800	17261.8470	8669.5849
830	142144.7900	154597.2600	10380.0770	8550.1649
831	147066.4000	155149.2900	15301.6870	9102.1949
832	149088.1600	155678.1900	17323.4470	9631.0949
833	143191.8600	155025.2100	11427.1470	8978.1149
834	144403.2900	155930.0100	12638.5770	9882.9149
835	146013.2900	155683.1200	14248.5770	9636.0249
836	146935.2600	156555.2700	15170.5470	10508.1749
837	147771.4400	156163.0400	16006.7270	10115.9449
838	148886.1800	156757.9800	17121.4670	10710.8849
839	149951.9000	157738.2800	18187.1870	11691.1849
840	150870.2500	156890.8400	19105.5370	10843.7449
841	138760.2800	153074.0100	6995.5670	7026.9149
842	138393.6800	154240.8900	6628.9670	8193.7949
843	139465.5100	154309.7300	7700.7970	8262.6349
844	137872.6200	155016.2100	6107.9070	8969.1149
845	139064.1200	155217.4300	7299.4070	9170.3349
846	138015.1800	155514.9600	6250.4670	9467.8649
847	140500.8400	155007.6800	8736.1270	8960.5849
848	138793.5200	155638.1900	7028.8070	9591.0949
849	139842.8200	155778.7000	8078.1070	9731.6049
850	138989.3200	156131.4500	7224.6070	10084.3549
851	139881.7800	156200.2800	8117.0670	10153.1849
852	137348.1400	155938.4600	5583.4270	9891.3649
853	139202.4500	156601.0100	7437.7370	10553.9149
854	141521.1700	156506.1000	9756.4570	10459.0049
855	143425.7500	157339.4000	11661.0370	11292.3049
856	144718.6800	157602.5600	12953.9670	11555.4649
857	137889.6800	157781.6600	6124.9670	11734.5649
858	136656.2300	157553.5500	4891.5170	11506.4549
859	139713.8100	158198.0500	7949.0970	12150.9549
860	142527.7600	158312.9700	10763.0470	12265.8749
861	144512.6700	158820.0300	12747.9570	12772.9349
862	135607.1000	159148.9200	3842.3870	13101.8249
863	142990.5500	159397.0900	11225.8370	13349.9949
864	144348.8600	159828.8200	12584.1470	13781.7249
865	143385.2200	160463.4800	11620.5070	14416.3849
866	141629.7100	160163.6500	9864.9970	14116.5549
867	140522.9900	160396.7500	8758.2770	14349.6549
868	137404.8000	160392.0300	5640.0870	14344.9349

869	139356.8500	160405.1700	7592.1370	14358.0749
870	145961.7700	157814.6700	14197.0570	11767.5749
871	147221.0600	157904.4100	15456.3470	11857.3149
872	149007.8800	158255.2300	17243.1670	12208.1349
873	147900.6400	158283.6700	16135.9270	12236.5749
874	150211.9400	158547.6400	18447.2270	12500.5449
875	145488.5900	158833.2500	13723.8770	12786.1549
876	148375.7200	159294.2600	16611.0070	13247.1649
877	149100.7800	159241.6700	17336.0670	13194.5749
878	145614.2600	159480.6800	13849.5470	13433.5849
879	150413.2200	159364.6600	18648.5070	13317.5649
880	146987.6800	159418.2100	15222.9670	13371.1149
881	145693.6500	160063.0700	13928.9370	14015.9749
882	148723.8400	160336.6900	16959.1270	14289.5949
883	146444.3400	161044.5800	14679.6270	14997.4849
884	149877.2200	161091.0400	18112.5070	15043.9449
885	156650.5300	160402.7300	24885.8170	14355.6349
886	154830.0900	160255.0500	23065.3770	14207.9549
887	155780.6100	161155.0600	24015.8970	15107.9649
888	155973.6900	161520.8100	24208.9770	15473.7149
889	155757.6200	161698.6300	23992.9070	15651.5349
890	153146.9700	160216.4700	21382.2570	14169.3749
891	155477.3700	162651.1300	23712.6570	16604.0349
892	164429.8700	163489.9700	32665.1570	17442.8749
893	158431.2500	162322.4000	26666.5370	16275.3049
894	158487.3400	164413.3800	26722.6270	18366.2849
895	154680.7500	164650.3300	22916.0370	18603.2349
896	169251.0600	165333.1600	37486.3470	19286.0649
897	165295.9600	166307.4800	33531.2470	20260.3849
898	157354.7000	166956.6600	25589.9870	20909.5649
899	163428.0200	166293.3500	31663.3070	20246.2549
900	161442.7300	165833.1700	29678.0170	19786.0749

附件 2 各区域货物运输量三维图



附件 3 圆覆盖模型代码（Python）

迭代函数：

```
import math
import numpy as np

def iteration_func(x0, y0, lamda, rest_set):
    '''

    :param x0:  初始点的横坐标

    :param y0:  初始点的纵坐标

    :param lamda:  步长

    :param rest_set: center 的子列表，所有未被放入覆盖圆（二级节点）的中心点集合

    :return: (x_new,y_new)迭代调整后圆的新圆心
    '''
    x = 0
    y = 0
    for item in rest_set:
        x += item[0]
        y += item[1]

    x = x - len(rest_set) * x0
    y = y - len(rest_set) * y0

    fenmu = math.sqrt(x**2 + y**2)

    alpha = math.acos(x/fenmu)
    beta = np.random.rand() * math.pi/36 * math.pow(-1, int(np.random.random_integers(1, 100, 1)))

    return (x0 + lamda/10 * math.cos(alpha + beta), y0 + lamda * math.sin(alpha + beta)* math.pow(-1, int(np.random.random_integers(1, 100, 1))))
    # return (x0 + lamda * math.cos(alpha + beta), y0 + lamda * math.sin(alpha + beta))
```

主函数：

```
from iteration_func import iteration_func
import pickle
from matplotlib import pyplot as plt

R = 3
def judge_func(current_circle_x, current_circle_y, judge_x, judge_y):
    if (current_circle_x - judge_x) ** 2 + (current_circle_y - judge_y) ** 2 <=
R ** 2:
        return True
    else:
        return False

with open('ceter_datas.pickle', 'rb') as e:
    center = pickle.load(e)
# print(len(center))
all_index = [i for i in range(94, len(center))]  # 建立所有中心点的索引列表，
方便后续在列表上索引
included_set = []  # 已经放入圆中（即被二级节点服务）的中心点集合，
索引放入
rest_set = [i for i in range(94, len(center))]  # 还未被服务的点的集合，同
上;初始为全集


original_x = 19
original_y = 11  # 初始化从个（0,0）开始搜索


xx = []
yy = []
count = 0
while count < 500:
    # print((original_x, original_y))
    xx.append(original_x)
    yy.append(original_y)
    temp_include = []
```

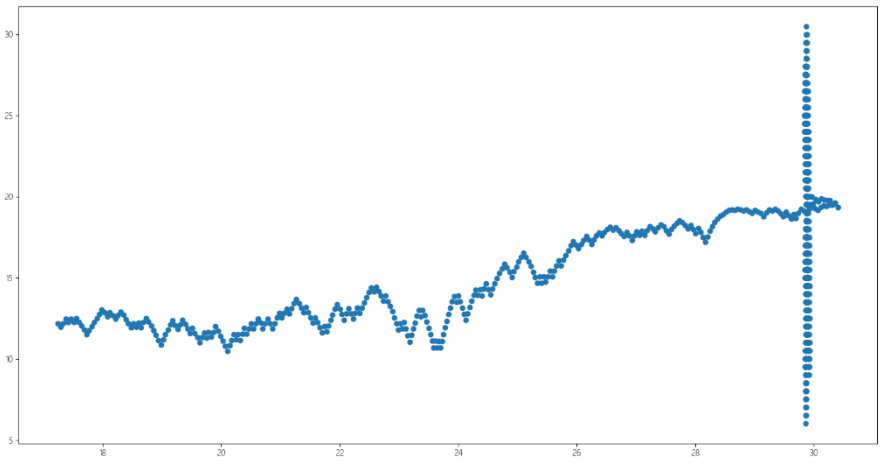
```
## 在未被覆盖的中心点中寻找所有可以被当前服务的中心点
for i in rest_set:
    if judge_func(original_x, original_y, center[i][0], center[i][1]):
        temp_include.append(i)
# print(temp_include)
## 计算当前区域内的点的货物量
cap = 0
for i in temp_include:
    cap += center[i][2]
if 2500 < cap <= 3000:
    included_set.extend(temp_include)

    for item in temp_include:
        rest_set.remove(item)
    print(temp_include, cap, (original_x, original_y))

else:
    para = []
    for i in rest_set:
        para.append(center[i])
    new_yuanxin = iteration_func(original_x, original_y, 0.5, para)
    original_x = new_yuanxin[0]
    original_y = new_yuanxin[1]
count += 1
```

附件 4 圆覆盖模型初始输出

[95, 99] 3576.82111679 (20.623761963574616, 12.485398539207521)
[94, 96, 97] 2907.76837737 (22.298598646017805, 13.162248712954264)
[98, 100, 102] 2582.16593691 (24.431732805930338, 14.315947981065996)
[103, 109] 2180.31178451 (27.23174651273536, 18.179523302196408)
[101, 108] 2970.05765829 (30.407634741899944, 19.368135328676303)



附件 5 最近邻改进算法代码（Python）

```
from iteration_func import iteration_func
import pickle
import math

R = 3
def judge_func(current_circle_x, current_circle_y, judge_x, judge_y):
    if (current_circle_x - judge_x) ** 2 + (current_circle_y - judge_y) ** 2 <=
R ** 2:
        return True
    else:
        return False

def distance_computer(cluster_point, target_point):
    return math.sqrt((cluster_point[0] - target_point[0])**2 + (cluster_point[1]
- target_point[1])**2)

with open('ceter_datas_tongshun.pickle', 'rb') as e:
    center = pickle.load(e)


all_index = [i for i in range(0, len(center))]   # 建立所有中心点（除去初始化

的 791 中心点）的索引列表，方便后续在列表上索引

clustered_set = []    # 已经放入圆中（即被二级节点服务）的中心点集合，

索引放入

rest_set = [i for i in range(0, len(center))]    # 还未被服务的点的集合，同上;

初始为全集，除掉第一个中心点



hahahaha = []

while len(rest_set) != 0:

    cluster_x = center[rest_set[0]][0]
    cluster_y = center[rest_set[0]][1]

    cluster_capacity = center[rest_set[0]][2]    # 初始化此次聚类的开始中

心点和初始类别载货量
```

```
        pop_point = rest_set.pop(0)

        inner_clustered_set = [pop_point]
        inner_rest_set = [i for i in rest_set]

        while cluster_capacity <= 3000:

            # inner_clustered_set = [pop_point]
            # inner_rest_set = [i for i in rest_set]


            ## 通过一次 for 循环，把离当前最近的点找出来

            temp = 100
            flag = 0
            for rest_i in inner_rest_set:
                dis = distance_computer((cluster_x, cluster_y), (center[rest_i][0],
center[rest_i][1]))
                if dis < temp:
                    temp = dis
                    flag = rest_i


            ## 临时更新质心

            temp_cluster_x = 0
            temp_cluster_y = 0
            inner_clustered_set.append(flag)
            for item in inner_clustered_set:
                temp_cluster_x += center[item][0]
                temp_cluster_y += center[item][1]
            temp_cluster_x = (temp_cluster_x + cluster_x) /
(len(inner_clustered_set) + 1)
            temp_cluster_y = (temp_cluster_y + cluster_y) /
(len(inner_clustered_set) + 1)


            ## 判断所有的点是否满足到质心距离小于 3

            bool_list = []
            for item in inner_clustered_set:
                bool_list.append(judge_func(temp_cluster_x, temp_cluster_y,
center[item][0], center[item][1]))
            # print(bool_list)
            c = 0
```

```
            out_set = []
            for item in bool_list:
                if item == 0:
                    out_set.append(c)
                c += 1


            ##  如果所有点都仍在新质心范围内

            if len(out_set) == 0:
                if cluster_capacity + center[flag][2] <= 3000:
                    inner_rest_set.remove(flag)
                    cluster_capacity += center[flag][2]
                    cluster_x = temp_cluster_x
                    cluster_y = temp_cluster_y
                    # print('***')
                else:
                    inner_clustered_set.remove(flag)
                    break


            ##  如果有部分中心点因为质心调整后溢出了

            elif len(out_set) >= 2:
                inner_clustered_set.remove(flag)

            else:
                if center[inner_clustered_set[out_set[0]]][2] >= center[flag][2]:
                    inner_clustered_set.remove(flag)
                else:
                    cluster_x = temp_cluster_x
                    cluster_y = temp_cluster_y
        for i in inner_clustered_set[1:]:
            rest_set.remove(i)

        print((cluster_x, cluster_y), inner_clustered_set, cluster_capacity)
        # hahahaha.extend(inner_clustered_set)
        # print()
        # print(hahahaha)
print(rest_set)
```

附件 6 最近邻改进算法原始输出

(13.878735746031754, 4.4145371142857046) [0, 3, 5, 12, 13, 4] 2445.28172685

(16.796360666666679, 4.3614353999999995) [1, 8, 14, 17] 2323.75742437

(10.428501111111123, 4.7354626777777655) [2, 6, 10, 11, 16] 2643.7350461

(8.3318630000000109, 5.4400468999999898) [7, 22, 23, 19] 2623.6931913

(12.045577805555562, 6.0302010666666632) [9, 15, 18, 20, 27] 2661.42257128

(18.38370797222224, 7.8525153166666604) [21, 34, 35, 38, 41] 2695.17737318

(10.306112833333342, 6.6674907333333246) [24, 26, 28] 2818.74564325

(13.238389861111118, 7.4735395666666671) [25, 30, 37, 29, 33] 2715.30027012

(15.383280333333348, 7.5514348999999932) [31, 40] 1967.63593904

(8.646611666666681, 8.0720363999999947) [32, 36, 52, 56] 2953.92718681

(10.955090666666683, 9.4080448999999966) [39, 42, 63, 43] 2929.24171949

(15.175527166666683, 10.432836233333328) [44, 45, 46, 80] 2672.80347333

(17.476707000000001, 11.037651566666668) [47, 48] 2925.5552297

(18.192249027777791, 12.364330705555554) [49, 83, 81, 88, 86] 2656.65319261

(6.7366600000000076, 8.2523168999999967) [50, 51, 53, 54] 2705.96037609

(6.7534486666666727, 9.6630640666666618) [55, 57, 59] 2602.05255405

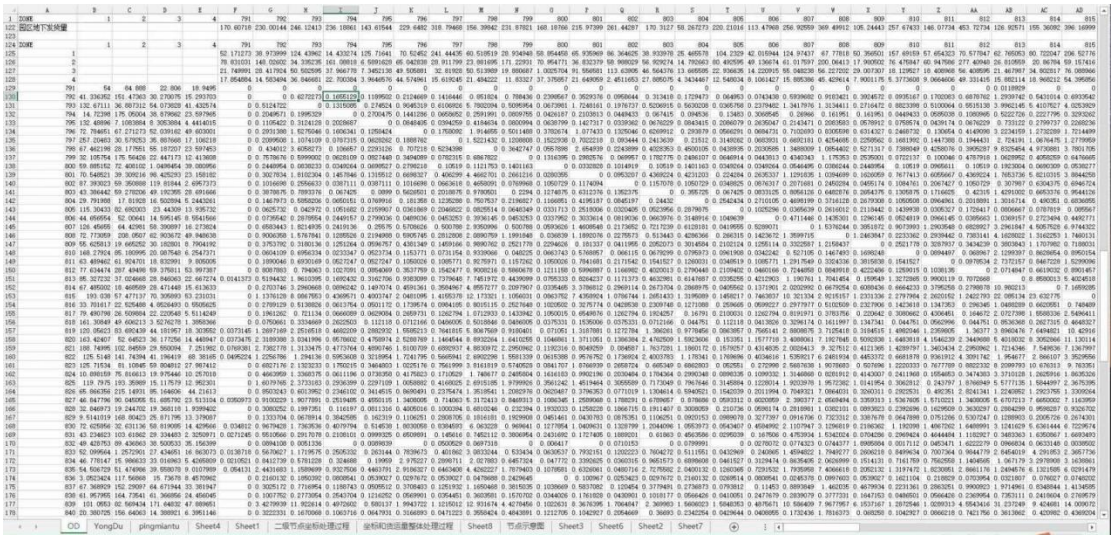
(7.601596138888902, 10.809616372222218) [58, 60, 62, 68, 66] 2544.19031969

(4.9749896666666826, 12.015384566666658) [61, 67, 71, 77] 2413.19733848

(11.908299222222235, 12.193987816666658) [64, 65, 70, 69, 72] 2551.44649985

(13.35579335317461, 13.880037689682537) [73, 74, 90, 87, 84, 92] 2879.14045723

附件 7 地下运输 OD 流量矩阵（部分）



附件 8 问题二相关代码（Matlab）

```
function [fa,shou] = compute_inner_outter_func(node, od)
    center_set = node;
    total_fa = 0;
    total_shou = 0;
    for center_index = 1:size(center_set,2)
        indexx = center_set(1,center_index);
        total_fa = total_fa + sum(od(indexx,:));
        total_shou = total_shou + sum(od(:,indexx));
    end
    fa = total_fa;
    shou = total_shou;
end



result_a3 = zeros(1,1);
for s = 0:5
    for e = 0:5
        if s ~= e
            result_a3(s+1,e+1) =
compute_function_fahuo(node3{s+1},node3{e+1},od_under);
        end
    end
end


function [ F ] = compute_function_fahuo(startzone,endzone,od)
    volume = 0;
    for startindex = 1:size(startzone,2)
        row = startzone(startindex);
        for endindex = 1:size(endzone,2)
            col = endzone(endindex);
            volume = volume + od(row,col);
        end
    end
    F = volume;
end


function [ F ] = computer_flow(startzone,endzone,od)
    volume = 0;
    for startindex = 1:size(startzone,2)
        row = startzone(startindex);
```

```
        for endindex = 1:size(endzone,2)
            col = endzone(endindex);
            volume +
        end
    end

end


function [julijuzhen] = jisuanjuli(zuobiao)
    [m,n] = size(zuobiao);
    distance = zeros(m,m);
    for i = 1:m
        start = zuobiao(i,:);
        sx = start(1,1);
        sy = start(1,2);

        for j = 1:m
            ends = zuobiao(j,:);
            ex = ends(1,1);
            ey = ends(1,2);
            distance(i,j) = sqrt((sx-ex)^2+(sy-ey)^2);

        end

    end
    julijuzhen = distance;
end
```